A Joint Standard of AASHTO, ITE, and NEMA

NTCIP 1102:2004 v01.15

National Transportation Communications for ITS Protocol

Octet Encoding Rules (OER) Base Protocol

October 2005

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

Published by

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

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

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

NOTICES

Copyright Notice

PDF File License Agreement

Data Dictionary and MIB Distribution Permission

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

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

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

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

Use of these materials does not constitute an endorsement or affiliation by or between AASHTO, ITE, or NEMA and the user, the user's company, or products and services of the user's company.

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

PRL and RTM Distribution Permission

To the extent that these materials are distributed by AASHTO / ITE / NEMA in the form of a Profile Requirements List ("PRL") or a Requirements Traceability Matrix ("RTM"), AASHTO / ITE / NEMA extend the following permission:

(i) you may make and/or distribute unlimited copies, including derivative works of the PRL (then known as a Profile Implementation Conformance Statement ("PICS")) or the RTM, provided that each copy you make and/or distribute contains the citation "Based on NTCIP 0000 [insert the document number] PRL or RTM. Used by permission. Original text (C) AASHTO / ITE / NEMA.";
(ii) you may not modify the PRL or the RTM except for the Project Requirements column, which is the only column that may be modified to show a product's implementation or the project-specific requirements; and
(iii) if the PRL or RTM excerpt is made from an unapproved draft, add to the citation "PRL (or RTM) excerpted from a draft document containing preliminary information that is subject to change."

This limited permission does not include reuse in works offered by other standards developing organizations or publishers, and does not include reuse in works-for-hire, compendiums, or electronic storage devices that are not associated with commercial hardware or software products intended for field installation.

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

Content and Liability Disclaimer

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

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

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

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

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

Trademark Notice

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

ACKNOWLEDGEMENTS

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

At the time that this document was prepared, the following individuals were the members of the NTCIP BSP WG:

- Robert De Roche (Chair)
- Bud Kent
- Alexis Mousadi
- Brian Paulsmeyer
- Joerg 'Nu' Rosenbohm
- Sonja Sun
- Kenneth Vaughn
- Hoi Wong

Other individuals providing input to the document include:

- Blake Christie
- Ken Earnest
- Gary Meredith

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

- Caltrans
- Econolite Control Products, Inc.
- ITERIS Inc.
- Mitretek
- Ontario Ministry of Transport
- PB Farradyne, a Division of Parsons Brinckerhoff Quade Douglas, Inc.
- Peek Traffic Systems, Inc.
- Pennsylvania DOT
- Siemens ITS - Eagle Traffic Control Systems
- Southwest Research Institute (SWRI)
- VHB
- Virginia DOT
- U.S. Department of Transportation Federal Highway Administration

FOREWORD

This document defines a base standard for encoding rules that are to be used in conjunction with Application Layer protocols defined in separate standards. There is one informative annex to this document.

This document is an NTCIP Protocol Base Standard standards publication. Protocol Base Standards define the basic details of data handling. A Joint NTCIP Protocol Base Standard standards publication is equivalent to these document types at the standards organizations:

AASHTO – Standard Specification
ITE – Software Standard
NEMA – Standard

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

User Comment Instructions

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

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

A User Comment should be submitted to this address:

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

A User Comment should be submitted in the following form:

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

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

Approvals

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

AASHTO – Standard Specification; March 2002
ITE – Software Standard; December 2002
NEMA – Standard; November 2004

History

A subset of the encoding rules defined in this NTCIP 1102 standard was specified in the standard NEMA TS 3.2-1996, *NTCIP Simple Transportation Management Framework (STMF)*, which was also numbered NTCIP 1101. However, in order to address extended ASN.1 functionalities needed for center-to-center communications, the necessity to develop a stand-alone document became apparent. That need resulted in this standard, NTCIP 1102, which serves as the replacement for both Section 5.1.2.2 of NEMA TS 3.2-1996, and NEMA TS 3.2 Amendment 1 of 1998. The content of this standard is fully consistent with NEMA TS 3.2-1996 as amended by TS 3.2 Recommended Amendment 1 of 1998. However, NTCIP 1102 defines many additional features not contained in NEMA TS 3.2-1996 (NTCIP 1101).

The technical specification of NTCIP 1102 is identical to the former version, except as noted in the development history below:

NTCIP 1102 v01.06. July 1999 – Accepted as a User Comment Draft by the Joint Committee on the NTCIP. January 2000 – NTCIP Standards Bulletin B0040 distributed for user comment.

NTCIP 1102 v01.11. September 2000 – Accepted as a Recommended Standard. June 2001 – NTCIP Standards Bulletin B0062 referred v01.12 for balloting. Approved by AASHTO in March 2002, approved by ITE in December 2002, and approved by NEMA in November 2004.

NTCIP 1102 v01.13. June 2002 – The BSP WG incorporated revisions to dispose of a user comment received during the ballot period. Revisions were made to: clause 1.3.2 Other References (to add URLs); clause 2.3.8.2 Content, b. Root Components (to improve DEFAULT value clarity); figure 2-23 Encoding Example 2 (to correct the example of the Root Object 3's Content Octets); and clause 2.3.9.2 SEQUENCE OF Content (to revise the *quantity* field specification).

NTCIP 1102:2004 v01.15. October 2005 – Prepared document for publication with revision of the front matter and editing for format and clarity.

Compatibility of Versions

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

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

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

INTRODUCTION

The context of the NTCIP is one part of the Intelligent Transportation Systems standardization activities covering base standards, profiles, and registration mechanisms.

- Base Standards define procedures and rules for providing the fundamental operations associated with communications and information that is exchanged over fixed-point communications links.

- Profiles define subsets or combinations of base standards used to provide specific functions or services. Profiles prescribe particular subsets or options available in base standards necessary for accomplishing a particular function or service. This provides a basis for the development of uniform, nationally recognized conformance.

- Registration Mechanisms provide a means to specify and uniquely identify detailed parameters within the framework of base standards or profiles.

In 1992, the NEMA 3-TS Transportation Management Systems and Associated Control Devices Section began development of the NTCIP. The Transportation Section's purpose was in response to user needs to include standardized systems communication in the NEMA TS 2 standard, *Traffic Controller Assemblies*. Under the guidance of the Federal Highway Administration's NTCIP Steering Group, the NEMA effort was expanded to include the development of communications standards for all transportation field devices that could be used in an Intelligent Transportation Systems (ITS) network.

In September 1996, an agreement was executed among AASHTO, ITE, and NEMA to jointly develop, approve, and maintain the NTCIP standards. In 1998, the Joint AASHTO / ITE / NEMA Committee on the NTCIP chartered a working group to formalize the Octet Encoding Rules, as well as several other Base Standards. The first meeting of the working group was in January 1999.

It was decided early in the process not to invent new base protocols, but rather to research existing standards and to decide whether these would fulfill the requirements for the specific transportation environments for which they would be used. The NTCIP would only develop new standards when this research proved unfruitful.

The NTCIP Joint Committee's Base Standards and Protocol (BSP) Working Group was concerned with the identification of applicable standards to address a transportation need, the definition of base standards and protocols in case existing standards do not address the identified need, and their documentation in standards publications.

Due to overlap in WG membership and to achieve meeting efficiency, the BSP WG was merged with the Profiles WG. The merged Base Standards and Profiles WG is abbreviated BSP2 WG.

The NTCIP 1102 is a Presentation Layer base standard, in the 7-layer OSI Reference Model, for use in association with certain Application Layer protocols in center-to-roadside and center-to-center communications, such as the Simple Transportation Management Protocol (STMP) and DATEX-ASN. These encoding rules provide more compact transfer syntax than do the Basic Encoding Rules. These rules, however, still maintain the fully octet-aligned fields to enable efficient processing by modern computer systems.

There are several different sets of encoding rules, such as Basic Encoding Rules (BER) and Packed Encoding Rules (PER), that have been investigated but determined not to address the specific needs of certain Application Layer protocols (STMF and DATEX-ASN) used within the transportation community.

The objective is to facilitate the specification of ITS systems characterized by a high degree of interoperability and interchangeability of components.

CONTENTS

Page

Foreword ii
Introduction v

Section 1 GENERAL 1

1.1 Scope 1
1.2 Protocol and Layer Relationship 1
1.3 References 1
1.3.1 Normative References 2
1.3.2 Other References 2
1.4 Terms 4
1.5 Abbreviations and Acronyms 4

Section 2 CONFORMANCE 5

2.1 General Requirements 5
2.2 General Rules 5
2.2.1 Structure of an Encoding 5
2.2.2 Identifer Octets 5
2.2.3 Length Octets 8
2.2.4 Content Octets 9
2.3 Encoding of Types 9
2.3.1 Encoding of a Boolean Value 9
2.3.2 Encoding of an Integer value 10
2.3.3 Encoding of an Enumerated Value 13
2.3.4 Encoding of a Real Value 13
2.3.5 Encoding of a Bitstring Value 14
2.3.6 Encoding of an Octet String Value 16
2.3.7 Encoding of a Null Value 17
2.3.8 Encoding of a Sequence Value 17
2.3.9 Encoding of the Sequence of Type 21
2.3.10 Encoding of the Set Type 21
2.3.11 Encoding of the Set of Type 22
2.3.12 Encoding of the Choice Type 22
2.3.13 Encoding of the Object Identifier (OID) Type 23
2.3.14 Encoding of the Embedded-PDV Type 24
2.3.15 Encoding of the Restricted Character String Types 24
2.3.16 Encoding of the Unrestricted Character String Types 25
2.4 Encoding of Whole Numbers 25
2.4.1 Encoding of an Unsigned Integer 25
2.4.2 Encoding of a Signed Integer 25

Annex A JUSTIFICATION FOR OER 27

< This page is intentionally left blank. >

Section 1
GENERAL

1.1 SCOPE

This base protocol provides a set of encoding rules that can be applied to values of data structures defined using the ASN.1 notation (ASN.1 Types). The result of applying the rules is an unambiguous transfer syntax that may be used to transmit the data across an interface. The receiving station decodes the transfer syntax using these same rules.

At the time of publication, the Octet Encoding Rules (OER) have been developed for use with the Simple Transportation Management Protocol (STMP) and DATEX-ASN.

These encoding rules are intended for use at the time that a data exchange is required and are designed to provide a simple set of encoding rules for agencies that can be used on links with limited bandwidth.

1.2 PROTOCOL AND LAYER RELATIONSHIP

This base protocol specifies the Octet Encoding Rules, and it provides the rules to encode (prepare) data for transmission over various transfer services. Within the scheme of the ISO OSI 7-Layer Reference Model, OER is a Presentation Layer protocol, which may be used in conjunction with various Application Layer protocols. However, it has been developed specifically to fulfill the needs of the STMP and DATEX-ASN. Figure 1-1 shows the place of OER.

NTCIP Profiles	ISO Layers	Base Standard
APPLICATION PROFILE	APPLICATION LAYER	(not addressed by this standard)
	PRESENTATION LAYER	OER (this standard)
	SESSION LAYER	(not addressed by this standard)

Figure 1-1
OER - Base Protocol Relationship

1.3 REFERENCES

For approved revisions, contact:

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

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

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

1.3.1 Normative References

American National Standards Institute (ANSI)
11 West 42nd Street, 13th Floor
New York, NY 10036

ISO/IEC 8824-1: 1995	*Information technology—Abstract Syntax Notation One (ASN.1): Specification of basic notation*
ISO/IEC 8825-1:1995	*Information technology—ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)*
ISO/IEC 8825-2:1996	*Information technology—ASN.1 encoding rules: Specification of Packed Encoding Rules (PER)*

1.3.2 Other References

NTCIP 1101:1997	*NTCIP Simple Transportation Management Framework (STMF)*
NTCIP 1101 A1	*Amendment 1: NTCIP Simple Transportation Management Framework (STMF) – Amendment 1 (formerly TS 3.2-1996 – Amendment 1)*
NTCIP 1103 v01.01	*NTCIP Simple Transportation Management Protocol (STMP)*
NTCIP 2304 v01.06	*NTCIP Application Profile DATEX-ASN – Draft* (formerly: TS 3.AP-DATEX)
NTCIP 8003 v01.05	*NTCIP Profiles Framework*
NTCIP 9001:1999	*NTCIP Guide*

The following URLs contain information regarding encoding rules and their interaction with higher layer protocols (Application Layer protocols) for which they have been developed:

- *Basic Encoding Rules*, Vijay Mukhi's Technology Cornucopia, http://users.nec.com/vmis/ber.htm [Accessed June 15, 2000]
- *A Layman's Guide to a Subset of ASN.1, BER, and DER*, An RSA Laboratories Technical Note, Burton S. Kaliski Jr., Revised November 1, 1993, http://auchentoshan.cs.ucl.ac.uk:8877/htm/pkcs/layman.htm, [Accessed June 16, 2000]
- *Basic Encoding Rules*, Boots Cassel, February 27, 1996, http://www.csc.vill.edu/~cassel/netbook/ber/ber.html, [Accessed June 16, 2000]
- *Abstract Syntax and Transfer Syntax*, The Presentation Layer, David Smith, August 11, 1998, http://ganges.cs.tcd.ie/4ba2/presentation/syntaxandrep.html, [Accessed June 16, 2000]

1.4 TERMS

For the purpose of this Standard, the following terms apply:

Application Layer	That portion of the OSI Reference Model (Layer 7) that provides access to the communications services.
data	Information before it is interpreted.
datagram	A self-contained unit of data transmitted independently of other datagrams.
end-application	A process or program using the communications stack.
end system	The source or destination of an information exchange.
extensions	The term used within this document that refers to Extension Additions as used within ISO 8824-1.
host	(Internet usage) The physical and/or logical part of the end-system's application. A computer attached to one or more networks that supports users and runs application programs.
Intelligent Transportation Systems	A major national initiative to apply information, communication and control technologies in order to improve the efficiency of surface transportation.
Internet	A work-wide collection of connected networks running the Internet suite of protocols.
internetwork	The ability of devices to communicate across multiple networks.
network	A collection of subnetworks connected by intermediate systems and populated by end systems.
octet	An ordered sequence of eight bits, and when in an octet string, aligned on an octet boundary.
octet boundary	A dividing point in an octet string between one sequence of eight bits and the next sequence of eight bits.
octet string	A sequence of zero, one or more sequences of eight bits.
Open Systems Interconnection	An international effort to facilitate communications among computers of different manufacture and technology.
OSI Reference Model	A widely accepted structuring technique that provides an abstract representation of the communication process that is divided into seven basic, functional layers.
Presentation Layer	That portion of an OSI Reference Model (Layer 6) responsible for converting and organizing data from one format to another.
Session Layer	That portion of an OSI Reference Model (Layer 5) which manages a series of data exchanges between end-system applications.
Tag	A type denotation, which is associated with every ASN.1 type.

Transport Layer That portion of an OSI Reference Model (Layer 4) which attempts to guarantee reliable data transfer between two end-systems, using flow control and error recovery, and may provide multiplexing.

1.5 ABBREVIATIONS AND ACRONYMS

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

AASHTO	American Association of State Highway and Transportation Officials
ANSI	American National Standards Institute
ASCII	American National Standard Code for Information Interchange
ASN.1	Abstract Syntax Notation One
BER	Basic Encoding Rules
IAB STD	Internet Architecture Board Standard
IP	Internet Protocol
ISO	International Organization for Standardization
ITE	Institute of Transportation Engineers
ITS	Intelligent Transportation Systems
NEMA	National Electrical Manufacturers Association
NTCIP	National Transportation Communications for ITS Protocol
OER	Octet Encoding Rules
OSI	Open Systems Interconnection
PER	Packed Encoding Rules
RFC	Request for Comments
PDU	Packet Data Unit
PDV	Presentation Data Value
TCP	Transmission Control Protocol
UCS	Universal Multiple Octet Coded Character Set
UDP	User Datagram Protocol

Section 2
CONFORMANCE

2.1 GENERAL REQUIREMENTS

If there is a requirement for an implementation to adhere to the behavior defined by this base protocol in an instance of communication, it shall follow the rules defined in this base protocol.

2.2 GENERAL RULES

The following subsections describe the general encoding rules used within OER to encode and decode data streams for different ASN.1 types.

2.2.1 Structure of an Encoding

An ASN.1 data value can be viewed as consisting of three parts:

a. IDENTIFIER OCTETS
b. LENGTH OCTETS
c. CONTENT OCTETS

Some or all of these parts shall be omitted in an OER encoding according to the following rules:

a. The IDENTIFIER OCTETS are omitted unless otherwise indicated in Clause 2.3.
b. The LENGTH OCTETS and CONTENT OCTETS are present unless otherwise indicated in Clause 2.3.

Figure 2-1 is a valid illustration of an OER encoding assuming that each part is present.

Figure 2-1
Conceptual Structure of an OER Encoding

NOTE—Other references sometimes call these three parts “Type, Length, Value.”

Example: If transmitting an INTEGER with a value of 2, the encoding might be in 3 fields under certain conditions. The first field, IDENTIFIER OCTETS, would indicate that the data type is an INTEGER. The second field, LENGTH OCTETS, would indicate the size of the field containing the CONTENT OCTETS. The third field, CONTENT OCTETS, contains the value of the INTEGER that is being encoded (in this case a value of 2).

2.2.2 Identifier Octets

Conceptually, the first field of an OER Encoding defines the IDENTIFIER OCTETS as shown in Figure 2-1. The IDENTIFIER OCTETS identify the data type of the data to follow. In most cases, the IDENTIFIER OCTETS are not present within data packets because both the receiving and the sending entity have knowledge of the ASN.1 structure transmitted. However, the IDENTIFIER OCTETS are present when exchanging the components contained within CHOICE or SET data types.

If present, the IDENTIFIER OCTETS shall be assembled according to the rules defined in the following subclauses.

ASN.1 assigns a class and a tag number for each data type. ASN.1 tag types have multiple levels of tagging. OER encodes the outer most level (see examples) under this information into the IDENTIFIER OCTETS. The Class indicates one of the four defined ASN.1 classes as defined in Table 2-1.

Table 2-1
ASN.1 Type Classes

ASN.1 Class	***Bit 7 (msb)***	***Bit 6***
UNIVERSAL	0	0
APPLICATION	0	1
CONTEXT SPECIFIC	1	0
PRIVATE	1	1

NOTE—A tag that is not identified as a Class is automatically "CONTEXT SPECIFIC."

In most cases, tag numbers are less than 63 (decimal), in which case they are encoded according to the rules defined in Section 2.2.2.1. If a tag number is equal to or greater than 63 (decimal), the IDENTIFIER OCTETS are encoded according to the rules defined in Section 2.2.2.2.

2.2.2.1 Single Octet Encoding

If the tag number is less than 63, the IDENTIFIER OCTETs shall be encoded as shown in Figure 2-2. The two high order bits, bits 7 and 6 (see b7 and b6 as shown below in Figure 2-2), shall encode the ASN.1 Class as shown in Table 2-1. The remaining bits shall encode the Tag Number as a 6-bit unsigned value.

NOTE—All figures are shown in network order, i.e., the first byte shown is transmitted first, and within a byte the bits are ordered bit 7 to bit 0.

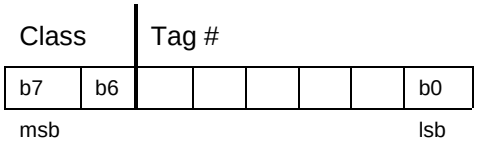

Figure 2-2
Structure of Single Octet Encoding of Identifier Octets

2.2.2.2 Multiple Octets Encoding

If the tag number is equal to or greater than 63, the IDENTIFIER OCTETS shall be encoded in multiple bytes as shown in Figure 2-3.

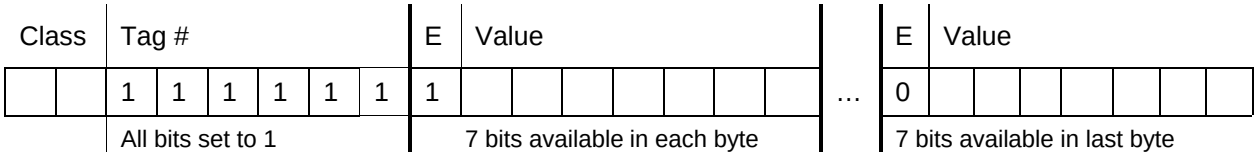

Figure 2-3
Structure of Multiple Octet Encoding of Identifier Octets

The 2 high order bits of the first byte shall encode the ASN.1 Class as shown in Table 2-1. The remaining bits of the first byte shall be set to 1.

The high order bit of the remaining IDENTIFIER OCTET bytes shall be the extension (E) bit. It shall be set to 1, except for the last byte in the IDENTIFIER OCTETS when it is set to 0. The lower seven bits of each remaining byte shall express the tag number as an unsigned binary integer.

The value of the second byte of the IDENTIFIER OCTET field shall not be 0x80.
NOTE—This ensures a minimum byte representation of the value, i.e., no leading zeros added.

2.2.2.3 Examples

A list of identifier octet values is shown in Table 2-2. For a complete listing of the standard ASN.1 tag numbers, please check ISO/IEC 8824 (ASN.1).

Table 2-2
Sample ASN.1 Tags

TYPE	***Resolved Type***	***Class***	***Tag Number***	***Identifier Octets***
INTEGER	INTEGER	UNIVERSAL	2	0x02
OCTET STRING	OCTET STRING	UNIVERSAL	4	0x04
NULL	NULL	UNIVERSAL	5	0x05
OBJECT IDENTIFIER	OBJECT IDENTIFIER	UNIVERSAL	6	0x06
SEQUENCE	SEQUENCE	UNIVERSAL	16	0x10
IpAddress	[APPLICATION 0] INTEGER	APPLICATION	0	0x40
Counter	[APPLICATION 1] INTEGER	APPLICATION	1	0x41
Gauge	[APPLICATION 2] INTEGER	APPLICATION	2	0x42
TimeTicks	[APPLICATION 3] INTEGER	APPLICATION	3	0x43
Opaque	[APPLICATION 4] OCTET STRING	APPLICATION	4	0x44
[3] INTEGER	[3] INTEGER	CONTEXT-SPECIFIC	3	0x83
[65] INTEGER	[65] INTEGER	CONTEXT-SPECIFIC	65	0xBF 41

NOTE—OER does not include a “constructor bit”, as used by BER. Thus, the identifier octets for SEQUENCE is 0x10 rather than 0x30.

A value of INTEGER:

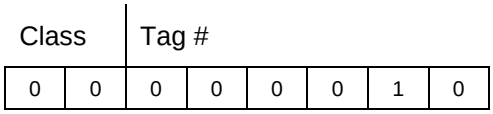

Figure 2-4
Identifier Octets Encoding - Integer

A value of SEQUENCE:

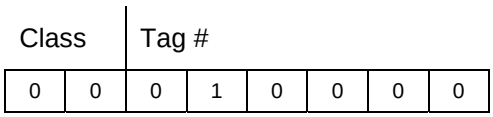

Figure 2-5
Identifer Octets Encoding - Sequence

A value of Gauge, which is an APPLICATION specific type (SNMP-specific):

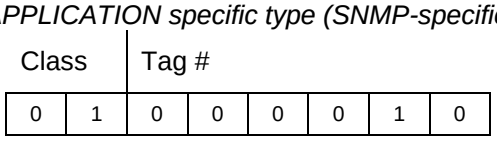

Figure 2-6
Identifier Octets Encoding - Gauge

2.2.3 LENGTH OCTETS

When present, the OER LENGTH OCTETS shall be encoded according to the rules defined in Subclauses 2.2.3.1 through 2.2.3.3.

2.2.3.1 Single Octet LENGTH OCTETS Encoding

If the CONTENT OCTETS contain fewer than 128 octets, the LENGTH OCTETS shall be encoded according to the short form as defined in Subclause 8.1.3.4 of BER.

NOTE—This Subclause states that the first high-order bit is set to 0 while the other 7 bits are used to express the value of the LENGTH OCTET.

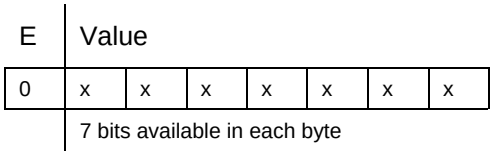

Figure 2-7
Single Octet Length Octets Encoding

Example: a length of 127 is encoded as:

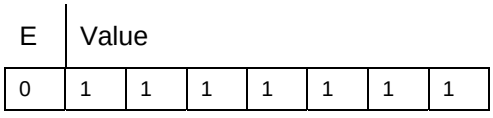

Figure 2-8
Single Octet Length Octets Encoding - Example: 127

2.2.3.2 Multiple Octet LENGTH OCTETS Encoding

If the CONTENT OCTETS contain 128 octets or more, the LENGTH OCTETS shall be encoded according to the long form as defined in Subclause 8.1.3.5 of BER, with the additional restriction that the second octet shall not have the value of zero.

NOTE—This Subclause states that the high-order bit of the high-order byte is set to 1 while the other 7 bits are used to express the number of bytes to follow.

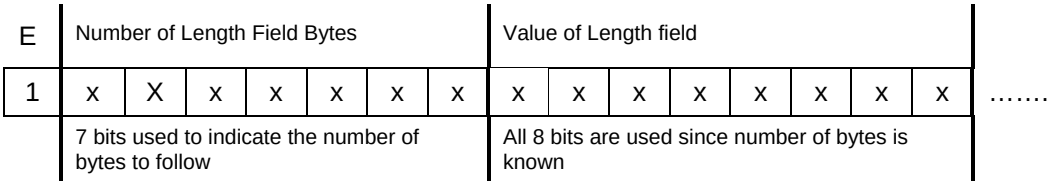

Figure 2-9
Multiple Octet Length Octets Encoding

Example: a length of 132 is encoded as

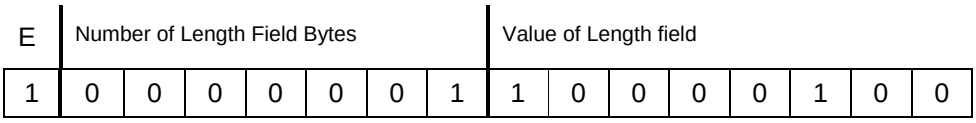

Figure 2-10
Multiple Octet Length Octets Encoding - Example: 132

2.2.3.3 Indefinite Octet LENGTH OCTETS Encoding

This form is reserved for future use. The values of 0x80 and 0xFF for the first LENGTH OCTET byte shall be reserved for this purpose.

2.2.4 Content Octets

The CONTENT OCTETS may consist of zero, one or more octets. The format of the CONTENT OCTETS are dependent upon the type being encoded; the proper format for each type is specified in Clause 2.3.

2.3 Encoding of Types

The following subsections describe the various rules for different ASN.1 types.

2.3.1 Encoding of a BOOLEAN value

NOTE—The IDENTIFIER OCTET for this type shall be omitted, unless this type is a component nested within a SET (Subclause 2.3.10) or CHOICE (Subclause 2.3.12) type.

2.3.1.1 Length

The LENGTH OCTETS for this type shall not be present.

2.3.1.2 Content

There shall be a single CONTENT OCTET. The CONTENT OCTET shall be zero when the Boolean value is FALSE. The CONTENT OCTET shall be any non-zero value when the Boolean value is TRUE.

2.3.2 Encoding of an INTEGER value

NOTE—The IDENTIFIER OCTET for this type shall be omitted, unless this type is a component nested within a SET (Subclause 2.3.10) or CHOICE (Subclause 2.3.12) type.

The encoding of an integer type shall be dependent on the presence of any range constraints, e.g., INTEGER (0..255) including any possible future extensions based on the presence of an extension marker, e.g., INTEGER (0..255, …).

2.3.2.1 Integer with Non-negative Values

If the range constraint does **not** allow any negative values, the following rules shall apply.

2.3.2.1.1 Non-negative 1-byte Integer

This subclause applies to any integer with a constraint specification that includes only non-negative values and:

a. min value: equal to or greater than 0 AND
b. max value: less than or equal to 255.

2.3.2.1.1.1 Length

The LENGTH OCTETS for this type shall not be present.

2.3.2.1.1.2 Content

The CONTENT OCTETS shall be encoded as a one-byte unsigned integer (see Section 2.4.1).

2.3.2.1.2 Non-negative 2-byte Integer

This subclause applies to any integer with a constraint specification that includes only non-negative values, can not be satisfied by Subclause 2.3.2.1.1 above, and:

a. min value: equal to or greater than 0 AND
b. max value: less than or equal to 65,535.

2.3.2.1.2.1 Length

The LENGTH OCTETS for this type shall not be present.

2.3.2.1.2.2 Content

The CONTENT OCTETS shall be encoded as a two-byte unsigned integer (see Section 2.4.1).

2.3.2.1.3 Non-negative 4-byte Integer

This subclause applies to any integer with a constraint specification that includes only non-negative values, can not be satisfied by Subclauses 2.3.2.1.1 and 2.3.2.1.2 above, and:

a. min value: equal to or greater than 0 AND
b. max value: less than or equal to 4,294,967,295.

2.3.2.1.3.1 Length

The LENGTH OCTETS for this type shall not be present.

2.3.2.1.3.2 Content

The CONTENT OCTETS shall be encoded as a four-byte unsigned integer (see Section 2.4.1).

2.3.2.1.4 Non-negative Unrestricted Integer

2.3.2.1.4.1 Length

The LENGTH OCTETS for this type shall be present.

2.3.2.1.4.2 Content

The CONTENT OCTETS shall be encoded as an unrestricted unsigned integer (see Section 2.4.1).

2.3.2.2 Integer including Negative Values

If the range constraint allows negative values, the following rules shall apply.

2.3.2.2.1 Negative 1-byte Integer

This subclause applies to any integer with a constraint specification that includes negative values and:

a. min value: equal to or greater than -128 AND
b. max value: less than or equal to 127.

2.3.2.2.1.1 Length

The LENGTH OCTETS for this type shall not be present.

2.3.2.2.1.2 Content

The CONTENT OCTETS shall be encoded as a one-byte signed integer (see Subclause 2.4.2).

2.3.2.2.2 Negative 2-byte Integer

This subclause applies to any integer with a constraint specification that includes negative numbers, can not be satisfied by Subclause 2.3.2.2.1 above, and:

a. min value: equal to or greater than -32768 AND
b. max value: less than or equal to 32,767.

2.3.2.2.2.1 Length

The LENGTH OCTETS for this type shall not be present.

2.3.2.2.2.2 Content

The CONTENT OCTETS shall be encoded as a two-byte signed integer (see Subclause 2.4.2).

2.3.2.2.3 Negative 4-byte Integer

This subclause applies to any integer with a constraint specification that includes negative values, can not be satisfied by Subclauses 2.3.2.2.1 and 2.3.2.2.2 above, and:

a. min value: equal to or greater than -2,147,483,648 AND
b. max value: less than or equal to 2,147,483,647.

2.3.2.2.3.1 Length

The LENGTH OCTETS for this type shall not be present.

2.3.2.2.3.2 Content

The CONTENT OCTETS shall be encoded as a four-byte signed integer (see Subclause 2.4.2).

2.3.2.2.4 Negative unrestricted Integer

If the conditions of Subclause 2.3.2.2.3 are not met, the following shall apply:

2.3.2.2.4.1 Length

The LENGTH OCTETS for this type shall be present.

2.3.2.2.4.2 Content

The CONTENT OCTETS shall be encoded as an unrestricted signed integer (see Section 2.4.2).

2.3.2.3 Examples

Table 2-3
Encoding of INTEGER Values—Examples

ASN.1 Type	Clause	Sample Value	Encoding (in hex)
INTEGER[1]	2.3.2.2.4	120	0x01(L) 0x78
Counter[2]	2.3.2.1.3	120 12345678	0x00 0x00 0x00 0x78 0x00 0xBC 0x61 0x4E
TimeTicks[2]	2.3.2.1.3	120 12345678	0x00 0x00 0x00 0x78 0x00 0xBC 0x61 0x4E
Gauge[2]	2.3.2.1.3	120 12345678	0x00 0x00 0x00 0x78 0x00 0xBC 0x61 0x4E
INTEGER (0..MAX)[3]	2.3.2.1.4	120	0x01(L) 0x78
INTEGER (0..255)	2.3.2.1.1	120	0x78
Counter (0..255)	2.3.2.1.1	120	0x78
INTEGER (0..2000)	2.3.2.1.2	120	0x00 0x78
INTEGER (1999..2000)	2.3.2.1.2	2000	0x07 0xD0
Gauge (1200..1250)	2.3.2.1.2	1200	0x04 0xB0
INTEGER (0..255, …)	2.3.2.2.4[4]	120	0x01(L) 0x78
INTEGER (-128..127)	2.3.2.2.1	120	0x78
INTEGER (-1000..1000)	2.3.2.2.2	-129	0xFF 0x7F
INTEGER {a(1), b(2)}	2.3.2.2.4[5]	3 [6]	0x01(L) 0x03
INTEGER {a(1), b(2)} (0..65535)	2.3.2.1.2[7]	3 [8]	0x00 0x03
INTEGER (-128..127) (0..MAX)	2.3.2.1.1	12	0x0C
INTEGER (-128..127) (0..MAX)	2.3.2.1.1	-128	invalid[9]

[1] No minimum or maximum values are specified for range as defined in RFC 1155

[2] IMPLICIT INTEGER (0..4294967295) as defined in RFC 1155

[3] Where MAX is defined as INTEGER (0.. 4294967295)

[4] Due to the extension marker

[5] STMP maps any object with this syntax to an ASN.1 ENUMERATED Type and the rule in subclause 2.3.3 apply.

[6] Both bits, Bit b0 and b1, set to 1

[7] Because the "NamedNumberList" is not significant according to ASN.1 rules, but the constraint would be visible.

[8] Both bits, Bit b0 and b1, set to 1

[9] The constraint (0..MAX) precludes this value.

2.3.3 Encoding of an ENUMERATED Value

NOTE—The IDENTIFIER OCTET for this type shall be omitted, unless this type is a component nested within a SET (Subclause 2.3.10) or CHOICE (Subclause 2.3.12) type.

2.3.3.1 Length

The LENGTH OCTETS for this type shall not be present.

2.3.3.2 Content

NOTE—The encoding of an ENUMERATED value is similar to the encoding of a LENGTH OCTET field except that the long form uses signed numbers rather than unsigned numbers.

If the value to be encoded is between zero (0) and 127, inclusive, the value shall be encoded as a one-byte unsigned integer (see Section 2.4.1).

If the value to be encoded is NOT within the range of zero (0) to 127, inclusive, the CONTENT OCTETS shall consist of two fields as follows.

- The first field shall be a one-byte unsigned integer (see Section 2.4.1) with a value equal to 128 plus the number of bytes contained in the second field.
- The second field within the CONTENT OCTETS shall be encoded as an unrestricted signed integer (see Section 2.4.1).

2.3.3.3 Examples

a. ENUMERATED {a(1), b(2), c(3)} the valid values would always be encoded in a single byte.
b. ENUMERATED {a(1), b(2), c(3), …} the currently defined values would be encoded in a single byte. If a future value is defined that is greater than 127, or less than 0, it would be encoded in more than one byte.
c. ENUMERATED {a(1), b(2), c(3), …, d(128)} the value 'd' would be encoded in three bytes with a representation as shown in Figure 2-11.

CONTENT OCTETS																								
First Field								Second Field																
E	Length							Value Byte 1								Value Byte 2								
1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	

Figure 2-11
Encoding of an ENUMERATED Value – EXAMPLE: 128

2.3.4 Encoding of a REAL Value

NOTE—The IDENTIFIER OCTET for this type shall be omitted, unless this type is a component nested within a SET (Subclause 2.3.10) or CHOICE (Subclause 2.3.12) type.

2.3.4.1 Length

NOTE—The LENGTH OCTETS will be present.

2.3.4.2 Content

The CONTENT OCTETS to encode a value represented in the ASN.1 data type “real” shall be as defined in ISO/IEC 8825-1, Subclause 8.5.6 (BER for CER/DER encoding).
NOTE—This Subclause states that all numbers are encoded as an ASCII string of the general form (+/-##.# e +/-##).

2.3.4.3 Examples

a. A value of “3.14” could be encoded as:

Length Octet	Content Octet (Byte 1)	Content Octet (Byte 2)	Content Octet (Byte 3)	Content Octet (Byte 4)
0x04	0x33	0x2E	0x31	0x34

Figure 2-12
Encoding of a Real Value – EXAMPLE: 3.14

b. A value of “2.345e12” could be encoded as:

Length Octet	Content Octets							
	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8
0x08	0x32	0x2E	0x33	0x34	0x35	0x65	0x31	0x32

Figure 2-13
Encoding of a Real Value – EXAMPLE: 2.345E12

2.3.5 Encoding of a BITSTRING Value

NOTE—The IDENTIFIER OCTET for this type shall be omitted, unless this type is a component nested within a SET (Subclause 2.3.10) or CHOICE (Subclause 2.3.12) type.

NOTE—The presence of a NamedBitList shall not restrict the values for a bit string type, meaning that even though only a limited number of bits may be defined within a NamedBitList, other bits may be reserved for future use. It cannot be assumed that a CONTENT OCTET of a BITSTRING type will automatically be encoded within the smallest number of bytes possible; e.g., if only 2 items are defined in a NamedBitList, it cannot be assumed that this value is encoded within 1 byte. Rather, without any ASN-visible constraint associated with this data type, a LENGTH OCTET needs to be encoded too.

If the bit string has a size constraint that does not contain an extension marker and the size of the BITSTRING is fixed to one value (e.g., SIZE (8)), the following rules apply.

- The LENGTH OCTETs shall be omitted.
- The CONTENT OCTETs shall be a bit string of length ‘X’ (taken from SIZE (X)) encoded according to Subclauses 8.6.2.1 of ISO/IEC 8825-1.
  NOTE—This Subclause states that the "first bit" (called bit 0) is encoded in the high order bit of the first byte and that the "trailing bit" follows in its relative position to the "first bit." In OER the initial octet indicating the number of unused bits shall not be used if the bitstring is of a fixed size.
- All remaining bits (i.e., the final bits to the last byte boundary) shall be set to zero

Bit Number of CONTENTS OCTETs

b0	b1	b2	b3	b4	b5	b6	b7	b8	b9	b10	b11	b12	b13	b14	b15
First Bit															Trail -ing Bit

Figure 2-14
Encoding of a Bitstring Value – Bit-Numbering Within 2 Bytes

If the above requirements do not apply, the LENGTH OCTETS shall be present and the CONTENT OCTETS shall be as defined in clause 8.6 of ISO/IEC 8825-1.

2.3.5.1 Examples

a. BITSTRING (SIZE (12)) with Bit 3 set would be encoded as.

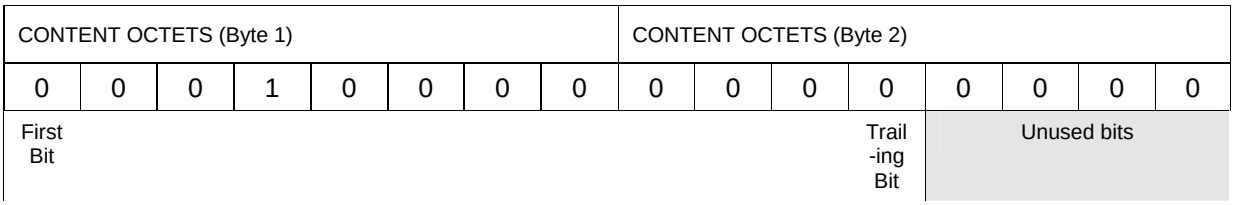

Figure 2-15
Encoding of a Bitstring Value – Example 1

b. BITSTRING (SIZE (8..32)) with Bit 3 set and the bitstring containing 20 bits would be encoded as

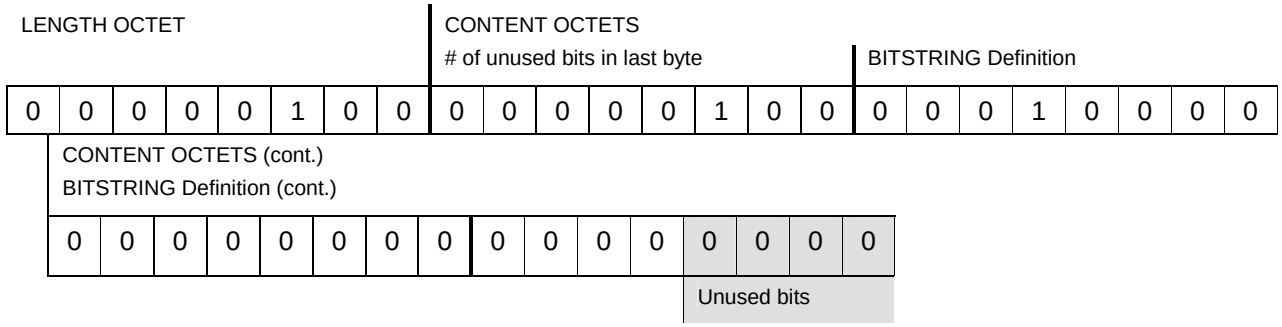

Figure 2-16
Encoding of a Bitstring Value – Example 2

c. BITSTRING (SIZE (8..32)) with Bit 3 set and the bitstring containing 14 bits would be encoded as

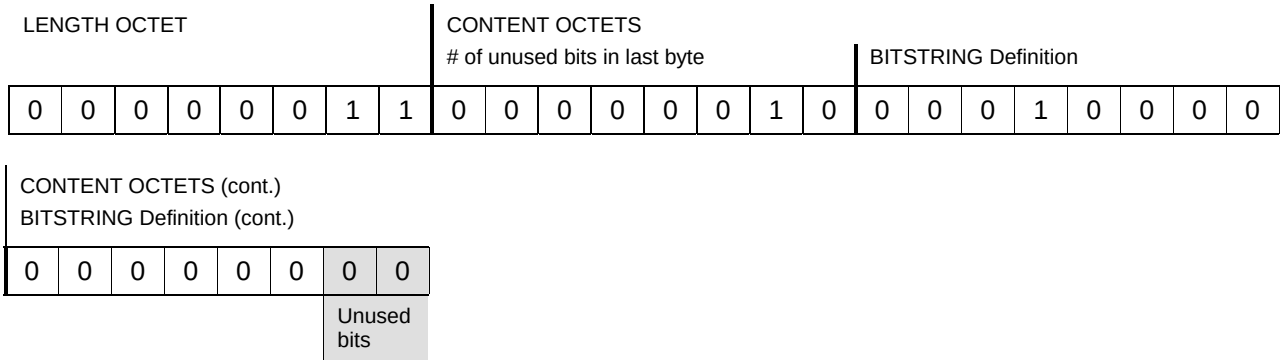

Figure 2-17
Encoding of a Bitstring Value – Example 3

d. BITSTRING (SIZE (8..32)) with Bit 13 set and the bitstring containing 14 bits would be encoded as

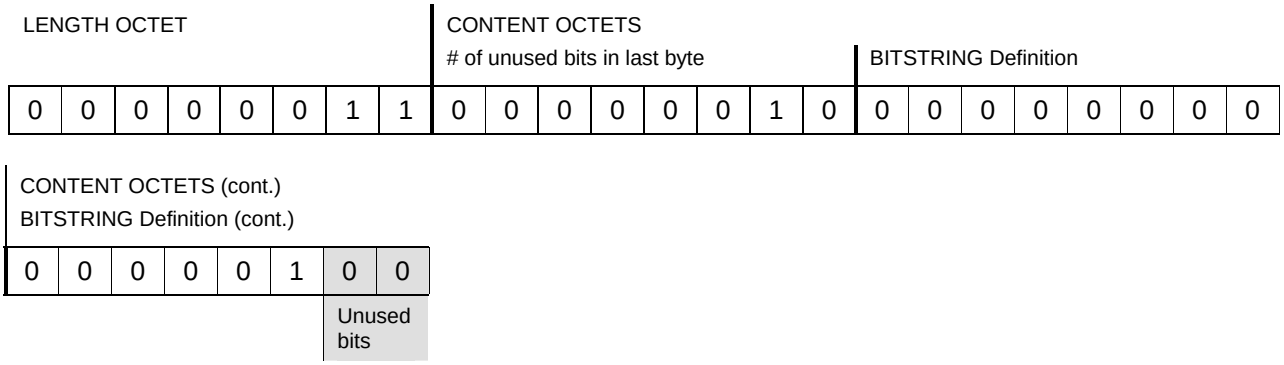

Figure 2-18
Encoding of a Bitstring Value – Example 4

e. BITSTRING with Bit 3 set and the bitstring containing 20 bits would be encoded as

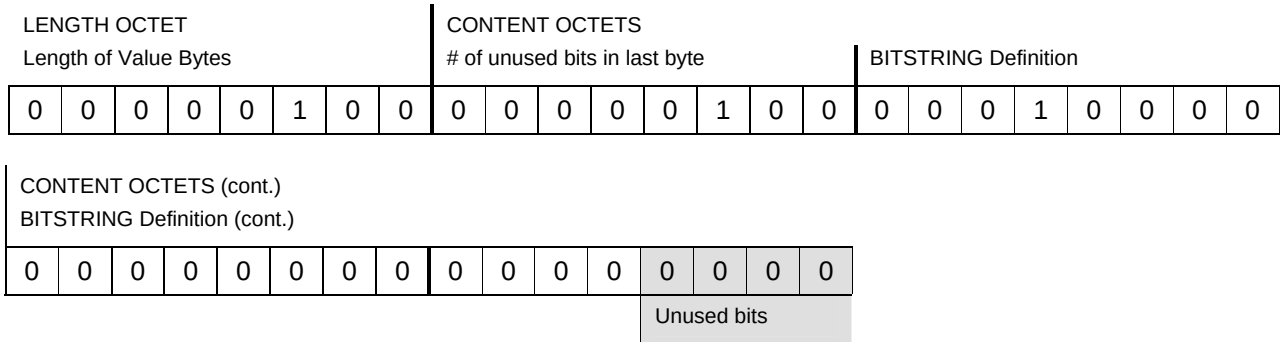

Figure 2-19
Encoding of a Bitstring Value – Example 5

f. BITSTRING (SIZE (0)): IDENTIFIER OCTET, LENGTH OCTET and CONTENT OCTET are omitted.

2.3.6 Encoding of an OCTET STRING Value

NOTE—The IDENTIFIER OCTET for this type shall be omitted, unless this type is a component nested within a SET (Subclause 2.3.10) or CHOICE (Subclause 2.3.12) type.

2.3.6.1 Length

If there is a size constraint that constrains all values of the OCTET STRING to the same length (i.e., the octet string must be exactly **X** octets), the LENGTH OCTETS shall be omitted; otherwise, the LENGTH OCTETS shall be present.

2.3.6.2 Content

The CONTENT OCTETs shall be as defined in Subclause 8.7.2 of ISO/IEC 8825–1 (BER).

2.3.6.3 Examples

a. OCTET STRING (SIZE (0..5)) would be encoded with a LENGTH OCTET field containing a value of 0-5 and CONTENT OCTET field containing that many octets. An OCTET STRING (SIZE (0..5))

containing the ASCII string “NTCIP“ would be encoded as:

Length Octet	Content Octets				
	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5
0x05	0x4E	0x54	0x43	0x49	0x50

Figure 2-20
Encoding of an Octet String Value – Example: Octet String (Size (0..5))

b. OCTET STRING (SIZE (5)) would be encoded with no LENGTH OCTET field and a CONTENT OCTET field containing exactly 5 octets.

Length Octet	Content Octets				
	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5
Not present	x	x	x	x	x

Figure 2-21
Encoding of an Octet String Value – Example: Octet String (Size (5))

c. OCTET STRING would be encoded with a LENGTH OCTET field and a CONTENT OCTET field containing that many bytes.

d. OCTET STRING (SIZE (0)): IDENTIFIER OCTET, LENGTH OCTET and CONTENT OCTET are omitted.

2.3.7 Encoding of a NULL Value

The IDENTIFIER OCTET, LENGTH OCTETs and CONTENT OCTETs shall be omitted.

2.3.8 Encoding of a SEQUENCE Value

NOTE—The IDENTIFIER OCTET for this type shall be omitted, unless this type is a component nested within a SET (Subclause 2.3.10) or CHOICE (Subclause 2.3.12) type.

NOTE—Encoding of this data type is similar to that as defined in ISO/IEC 8825-2, Clause 18, except the encoding of the fields follow the rules of OER.

2.3.8.1 Length

The LENGTH OCTETs for this type shall not be present.

2.3.8.2 Content

The CONTENT OCTETs shall consist of four fields in the following order: Preamble, Root, Extension Bits, Extension.

a. The Preamble shall be encoded as a BIT STRING of size X (i.e., BIT STRING (SIZE (X)). X is defined as:

   X = e + o
   e = 1, if Extension Marker is present, or 0, if Extension Marker is not present
   o = the number of OPTIONAL and DEFAULT components in the root

   NOTE—BIT STRING encoding is defined in Section 2.3.5. This encoding ensures an octet aligned field.

   If an Extension Marker is present, the first bit in the “Preamble“ shall indicate if any extensions are present in the encoding. If extensions are present, the bit shall be set to 1, otherwise the bit shall be set to 0.

   Each bit of the remaining BIT STRING shall indicate which, if any, of the OPTIONAL or DEFAULT components in the root are present. The bits shall be encoded in order with the first (remaining) bit indicating whether the first OPTIONAL or DEFAULT component is present, and the last bit indicating whether the last OPTIONAL or DEFAULT component is present. A bit shall be set to 1, if the associated component is present, otherwise, it shall be set to 0.

b. The Root field shall consist of a series of component fields, which encode the values of the RootComponents that are present. The RootComponents are those components defined in the RootCommonentTypeList. The component fields shall be encoded in order with the first field encoding the value of the first component present, and the last field encoding the value of the last component present. Any component with a DEFAULT value shall not be present if the value of the component is the DEFAULT value. Each component field shall consist of the OER encoding of the subject component.

   If the sequence does not contain an extension marker or if the extension marker bit (i.e., the first bit in the Preamble) is zero, the following fields (Extension Bits and Extension) are not present.

c. The Extension Bits field shall consist of an unsized OER-encoded BIT STRING that indicates which of the extensions are present. The bits shall be encoded in order with the first bit indicating whether the first extension is present, and the last bit indicating whether the last extension is present. A bit shall be set to 1, if the associated extension is present, otherwise, it shall be set to 0.

d. The Extension field shall consist of a series of extension component fields, which encode the values of the extensions that are present. The values shall be encoded in order with the first value indicating the value of the first present extension, and the last value indicating the value of the last present extension.

Each Extension Component field shall consist of an OCTET STRING. If the Extension is a ComponentType, the OCTET STRING shall contain the OER encoding of that value. If the Extension is an ExtensionAdditionGroup, the ExtensionAdditionGroup shall be encoded as if it were a SEQUENCE type, and the resulting encoding shall define the contents of the OCTET STRING.

NOTE—This has the effect of adding a LENGTH OCTET field in front of the subject encoding.

2.3.8.3 Example

a. SEQUENCE {objectName1 OCTET STRING (SIZE (5)), --value: NTCIP
objectName2 INTEGER} --value: 5

PREAMBLE	ROOT	Extension Bits	Extension
Not present	↓	Not present	Not present

Object 1		Object 2	
Length Octet	Content Octets	Length Octet	Content Octets
Not present	0x4E 0x54 0x43 0x49 0x50	0x01	0x05

Figure 2-22
Encoding of a Seqeunce Value – Example 1

Since there is neither an Extension Marker nor an OPTIONAL or DEFAULT value in this SEQUENCE, the PREAMBLE has a length of zero (0). The Root is composed of two component fields: the first indicates the size-constrained contents of the OCTET STRING, the second indicates the length and contents of the non-size-constrained INTEGER.

b. SEQUENCE {objectName1 OCTET STRING (SIZE (5)), --value: NTCIP
objectName2 INTEGER (0..255) DEFAULT 7, --value: 5
objectName3 INTEGER OPTIONAL --value: 255
}

PREAMBLE	ROOT	Extension Bits	Extension
11 000000	↓	Not present	Not present

Object 1		Object 2		Object 3	
Length Octet	Content Octet	Length Octet	Content Octets	Length Octet	Content Octets
Not present	0x4E 0x54 0x43 0x49 0x50	Not present	0x05	0x02	0x00 0xFF

Figure 2-23
Encoding of a Seqeunce Value – Example 2

This example contains no Extension Marker, but an OPTIONAL and a DEFAULT value.
The first 2 bits of the PREAMBLE (first byte) will indicate the presence of the OPTIONAL and the DEFAULT items. The remaining bits of this PREAMBLE byte will be set to zero (0).
The Root is composed of three component fields:

- the first indicates the size-constrained contents of the OCTET STRING,
- the second indicates the non-DEFAULT contents of the size-constrained DEFAULT INTEGER, and
- the third indicates the length and contents of the non-size-constrained existing OPTIONAL INTEGER.

c. SEQUENCE {objectName1 OCTET STRING (SIZE (5)), --value: NTCIP
…., --ExtensionMarker
…., --ExtensionMarker
objectName2 INTEGER} --value: 5

PREAMBLE	ROOT	Extension Bits	Extension
0000 0000	↓	Not present	Not present

Object 1		Object 2	
Length Octet	Content Octets	Length Octet	Content Octets
Not present	0x4E 0x54 0x43 0x49 0x50	0x01	0x05

Figure 2-24
Encoding of a Seqeunce Value—Example 3

Since there is an Extension Marker in this SEQUENCE, the PREAMBLE is existing. However, since the Extension Marker is unused, the first bit of the PREAMBLE is set to a value of zero (0). Additionally, the other 7 bits are set to zero (0) since there are no DEFAULT or OPTIONAL components.
The Root is composed of two component fields:

- the first indicates the size-constrained contents of the OCTET STRING,
- the second indicates the length and contents of the non-size-constrained INTEGER.

d. SEQUENCE {objectName1 OCTET STRING (SIZE (5)), --value: NTCIP
…,
objectName4 BITSTRING (SIZE (8)) OPTIONAL, --value=00011000
objectName5 OCTET STRING (SIZE (4..10)), --value=TEST
…,
objectName2 INTEGER (0..255) DEFAULT 7, --value=5
objectName3 INTEGER --value=120
}

PREAMBLE	ROOT	Extension Bits	Extension
1100 0000	↓	0000 0010 0000 0110 1100 0000	(see below)

ObjectName 1		ObjectName 2		ObjectName 3	
Length Octet	Content Octets	Length Octet	Content Octets	Length Octet	Content Octets
Not present	0x4E 0x54 0x43 0x49 0x50	Not present	0x05	0x01	0x78

Extension

ObjectName 4		ObjectName 5	
Length Octet	Content Octets	Length Octet	Content Octets
Not present	0x18 (=0001 1000)	0x04	0x54 0x45 0x53 0x54

Figure 2-25
Encoding of a Sequence Value—Example 4

Since there is an Extension Marker in this SEQUENCE, the PREAMBLE is existing. And because the Extension Marker is used, the first bit of the PREAMBLE is set to a value of one (1). Additionally, the next bit is set to one (1) since there is 1 OPTIONAL/DEFAULT component existing in the ROOT. The other 6 bits are set to zero (0) since there are no additional DEFAULT or OPTIONAL components in the ROOT. The Root is composed of three component fields:

- the first indicates the size-constrained contents of the OCTET STRING,
- the second indicates the size-constrained contents of the DEFAULT value (which is different from the actual DEFAULT value of 7), and
- the third indicates the length and contents of the non-size-constrained INTEGER

The next field is the Extension Bits field, which is encoded as an UNSIZED BITSTRING. This means:

- the first byte of the BITSTRING indicates the number of bytes to follow,
- the second byte indicates who many bits in the last byte are unused, and
- the following byte(s) indicate through the bit settings whether a particular component is present (bit 0 corresponds to Extension Component 1)

Since two Extension Components are indicated and because both have valid values, the encoding of the Extension Bits field is as follows:

- the first byte has a value of 0x02 to indicate that 2 bytes are following to express the content of this BITSTRING,
- the next byte has a value of 0x06 to indicate that 6 bits of the last (the next) byte are unused.
- the next byte has the first 2 bits set to 1 to indicate that there are 2 Extension Components present (while all other bits are unused and therefore set to zero (0)).

The Extension field consists of 2 component fields:

- the first one indicating the size-constrained contents of the OPTIONAL BITSTRING, and
- the second indicates the length and contents of the OCTET STRING.

2.3.9 Encoding of the SEQUENCE OF Type

NOTE—The IDENTIFIER OCTET for this type shall be omitted, unless this type is a component nested within a SET (Subclause 2.3.10) or CHOICE (Subclause 2.3.12) type.

2.3.9.1 Length

The LENGTH OCTETS for the SEQUENCE OF type shall not be present.

2.3.9.2 Content

The CONTENT OCTETS shall consist of two fields: a *quantity* and a *value*. The *quantity* field shall be an non-negative unrestricted integer (see Subclause 2.3.2.1.4) with a value equal to the number of times the ComponentType is repeated within the value field. The *value* field shall consist of the encoding of each component, taken in order.

2.3.10 Encoding of the SET Type

NOTE—The IDENTIFIER OCTET for this type shall be omitted, unless this type is a component nested within a SET (Subclause 2.3.10) or CHOICE (Subclause 2.3.12) type.

2.3.10.1 Length

The LENGTH OCTETS for this type shall not be present.

2.3.10.2 Content

The SET data type shall be encoded as if it were a SEQUENCE type, except that the encoding of each component within the set shall include the IDENTIFIER OCTETS.

2.3.11 Encoding of the SET OF Type

NOTE—The IDENTIFIER OCTET for this type shall be omitted, unless this type is a component nested within a SET (Subclause 2.3.10) or CHOICE (Subclause 2.3.12) type.

2.3.11.1 Length

The LENGTH OCTETS for this type shall not be present.

2.3.11.2 Content

The SET OF type shall be encoded as if it were declared as a SEQUENCE-OF type.

2.3.12 Encoding of the CHOICE Type

NOTE—The IDENTIFIER OCTET for this type shall be omitted, unless this type is a component nested within a SET (Subclause 2.3.10) or CHOICE (Subclause 2.3.12) type.

2.3.12.1 Length

The LENGTH OCTETS shall not be present.

2.3.12.2 Content

The CONTENT OCTETS of the CHOICE type shall consist of 2 fields:

- The first field shall contain the IDENTIFIER OCTETS of the selected type.
  NOTE—ASN.1 does not allow the different choices within a CHOICE statement to be of the same type. However, tag types (numbering) or 'automatic tagging' may be applied.
- The second field shall contain the value encoding of the selected type.

2.3.12.3 Example

```
a. CHOICE      {
                objectNameA    INTEGER,      --value=13
               objectNameB    INTEGER,      --value=14
               objectNameC    INTEGER       --value=15
               }
```

The above example is invalid unless the module has AUTOMATIC TAGGING applied. If AUTOMATIC TAGGING is applied, the above statement is equivalent to the one given below. Alternatively, the statement may be explicitly defined as given below.

```
CHOICE         {
                objectNameA    [0] INTEGER,  --value=13
               objectNameB    [1] INTEGER,  --value=14
               objectNameC    [2] INTEGER   --value=15
               }
```

Assume that the choice is "objectNameB", then the encoding is as follows:

First Field	Second Field	
Identifier Octet	Length Octet	Content Octets
0x81	0x01	0x0E

Figure 2-26
Encoding of a Choice Value – Example 1

The data type 'CHOICE' itself is not encoded. The IDENTIFIER OCTETs of the selected component (i.e., the 'objectNameB' INTEGER) are encoded followed by its 'normal' encoding. In this case, the INTEGER is not range-constrained. Thus, both the Length Octet as well as the Content Octet of the INTEGER need to be encoded.

```
b. CHOICE     {
                  objectNameA     INTEGER,                  --value=13
                  objectNameB     INTEGER,                  --value=14
                  objectNameC     INTEGER,                  --value=15
                  objectNameD     CHOICE {
                        objectNameE   OCTET STRING,
                        objectNameF   BOOLEAN           --value= >0},
                  objectNameG    OBJECT IDENTIFIER}
```

The above example is invalid unless the module has AUTOMATIC TAGGING applied. If AUTOMATIC TAGGING is applied, the above statement is equivalent to the one given below. Alternatively, the statement may be explicitly defined as given below.

```
CHOICE        {
                  objectNameA     [0] INTEGER,              --value=13
                  objectNameB     [1] INTEGER,              --value=14
                  objectNameC     [2] INTEGER,              --value=15
                  objectNameD     [3] CHOICE {
                        objectNameE   [0] OCTET STRING,
                        objectNameF   [1] BOOLEAN       --value= >0},
                  objectNameG    [4] OBJECT IDENTIFIER}
```

First, the square bracket statements needed to be added to identify which ones of the INTEGERS are being encoded. Within 'nested' statement, the square-bracket enumerations starts again from zero. Assume that the choice is "objectNameF," Then the encoding is as follows:

First Field	Second Field	
	First Field (nested)	Second Field (nested)
Identifier Octet	Identifier Octet	Content Octets
0x83	0x81	0x01

Figure 2-27
Encoding of a Choice Value – Example 2

The data type 'CHOICE' itself is not encoded. The IDENTIFIER OCTETs of the selected component (i.e., the 'objectNameD' CHOICE) are encoded followed by its 'normal' encoding. In this case, this encoding is a nested choice. Therefore, it is encoded as the IDENTIFIER OCTETs of the nested selected component (i.e., the BOOLEAN) followed by its content octets.

2.3.13 Encoding of the OBJECT IDENTIFIER (OID) Type

NOTE—The IDENTIFIER OCTET for this type shall be omitted, unless this type is a component nested within a SET (Subclause 2.3.10) or CHOICE (Subclause 2.3.12) type.

2.3.13.1 Length

The LENGTH OCTETS for this type shall be present.

2.3.13.2 Content

The CONTENT OCTETS shall be as defined in ISO/IEC 8825-1, clause 8.19.
NOTE—This clause and its subclauses state that the LENGTH OCTET is always present, that the CONTENT OCTETS must be an ordered list of encodings of the subidentifiers. It also defines how to encode subidentifier values larger than 127. Additionally, it defines that the first 2 subidentifiers are encoded within 1 byte following the ((X*40)+Y)-formula (where X = value of 1st subidentifier and Y = value of 2nd subidentifier).

2.3.13.3 Example

a. OBJECT IDENTIFIER of { iso(1) org(3) dod(6) internet(1) private(4) enterprises(1) nema(1206) transportation(4) protocols(1) dynObjMgmt(3) dynObjDef(1) dynObjEntry(1) dynObjVariable(3)} would be encoded as:

Length Octet	Content Octets												
	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte 9	Byte 10	Byte 11	Byte 12	Byte 13
0x0D	0x2B	0x06	0x01	0x04	0x01	0x89	0x36	0x04	0x01	0x03	0x01	0x01	0x03

Figure 2-28
Encoding of an Object Identifier Value – Example

The first byte is a representation for the combination of {ISO ORG} following the formula defined in Subclause 8.19.4 and 8.19.5 of ISO/IEC 8825-1.

The value of 1206 for “nema” needed to be encoded in 2 bytes since it exceeded a value of 127.

2.3.14 Encoding of the EMBEDDED-PDV Type

NOTE—The IDENTIFIER OCTET for this type shall be omitted, unless this type is a component nested within a SET (Subclause 2.3.10) or CHOICE (Subclause 2.3.12) type.

The EMBEDDED-PDV type shall be encoded as the OER encoding of the type as defined in Clause 32.5 of ISO/IEC 8824-1 (ASN.1-Part 1). This clause defines the associated type for the value definition as follows:

```
SEQUENCE {
    identification                  CHOICE {
        syntaxes                    SEQUENCE {
            abstract                OBJECT IDENTIFIER,
            transfer                OBJECT IDENTIFIER},
        syntax                      OBJECT IDENTIFIER,
        presentation-context-id     INTEGER,
        context-negotiation         SEQUENCE {
            presentation-context-id INTEGER,
            transfer-syntax         OBJECT IDENTIFIER},
        transfer-syntax             OBJECT IDENTIFIER,
        fixed                       NULL},

    string-Value                    OCTET STRING}
```

2.3.15 Encoding of the RESTRICTED CHARACTER STRING Types

NOTE—The IDENTIFIER OCTET for this type shall be omitted, unless this type is a component nested within a SET (Subclause 2.3.10) or CHOICE (Subclause 2.3.12) type.

The CONTENT OCTETS of the RESTRICTED CHARACTER STRING types shall be encoded according to the rules defined in Clause 8.20 of ISO 8825-1. This clause defines that values of this data type are encoded as an OCTET STRING.

2.3.16 Encoding of the UNRESTRICTED CHARACTER STRING Types

NOTE—The IDENTIFIER OCTET for this type shall be omitted, unless this type is a component nested within a SET (Subclause 2.3.10) or CHOICE (Subclause 2.3.12) type.

The CONTENT OCTETS of the UNRESTRICTED CHARACTER STRING types shall be encoded exactly as defined for the EMBEDDED-PDV type. This type is defined in Clause 39.5 of ISO/IEC 8824-1 (ASN.1-Part 1).

2.4 ENCODING OF WHOLE NUMBERS

2.4.1 Encoding of an Unsigned Integer

One-byte unsigned integer values shall be encoded in a bit-field of eight bits according to the rules defined in 10.3.2 - 10.3.5 of ISO/IEC 8825-2 (PER). NOTE—This equates to an 8-bit binary INTEGER.

Two-byte unsigned integer values shall be encoded in a bit-field of 16 bits according to the rules defined in 10.3.2 - 10.3.5 of ISO/IEC 8825-2 (PER). NOTE—This equates to a 16-bit binary INTEGER.

Four-byte unsigned integer values shall be encoded in a bit-field of 32 bits according to the rules defined in 10.3.2 - 10.3.5 of ISO/IEC 8825-2 (PER). NOTE—This equates to a 32-bit binary INTEGER.

Unrestricted unsigned integer values shall be encoded as a minimum octet non-negative-binary-integer as defined in 10.3.2 - 10.3.5 of ISO/IEC 8825-2 (PER).

2.4.2 Encoding of a Signed Integer

One-byte signed integer values shall be encoded as an eight bit 2's complement integer.

Two-byte signed integer values shall be encoded as a sixteen bit 2's complement integer.

Four-byte signed integer values shall be encoded as a thirty-two bit 2's complement integer.

Unrestricted signed integer values shall be encoded according to clause 8.3 of ISO/IEC 8825-1 (BER).

< This page is intentionally left blank. >

Annex A
JUSTIFICATION FOR OER

(Informative)

During the development of the NTCIP, in 1995, the NEMA 3-TS Technical Committee began to learn about the Simple Network Management Protocol (SNMP) and Abstract Syntax Notation One (ASN.1). The committee liked the design of both SNMP and ASN.1, but was concerned about the massive bandwidth requirements that the SNMP solution imposed. As such, the committee set forth to establish a new protocol, called the Simple Transportation Management Protocol (STMP), which would meet the bandwidth requirements of intelligent transportation systems.

The STMP was designed around the concept of configuring message content at run-time, but prior to actual message use, and using more compact encoding rules. Unfortunately, the NEMA committee was unaware of ISO's development of Packed Encoding Rules (ISO PER), ISO 8825-2. Thus, instead of using ISO PER, NEMA developed their own encoding rules, which were limited to the needs of STMP. To further complicate the matter, NEMA also named their encoding rules Packed Encoding Rules (NEMA PER).

By 1998, several firms were implementing the STMP and the supporting encoding rules, and through this process, several ambiguities were noticed in the NEMA PER standard. Further, the Committee realized that some people were becoming confused by the two versions of PER. Finally, the committee was being asked to investigate an extension to STMP that would use fixed messages. Thus, the Committee conducted an investigation to determine whether the standard should be modified to use ISO PER, or if the existing standard (STMP) should simply be clarified.

After a detailed review of the issue, the Committee, in association with various NTCIP Working Groups that had since been established, decided to clarify the original specification due to the inherent complexity of the ISO PER.

It also agreed that the long-term solution was to develop a separate document that would formally define these new encoding rules for all ASN.1 Types. Finally, it was decided that the name of the encoding rules should be changed in order to minimize confusion, and the committee decided upon the name "Octet Encoding Rules" because of the octet-aligned nature of the encoding. This document is the result of that effort.

However, there was concern expressed during the discussions that this issue would be raised in the future. Therefore, this Annex was added to document the key reasons the NTCIP Joint Committee decided to develop OER rather than using ISO PER. They are as follows, in order of significance:

1. By clarifying the existing standard (STMP), there would be minimal impact to the existing NEMA PER definition and no significant impact to current STMP implementations.
2. The committee felt that OER was easier to understand because it was based on byte-level encoding rather than bit-level encoding, which is the format in which most systems engineers think about data. This was deemed significant in that it would facilitate design, programming, debugging, and systems integration.
3. The proposal for the OER document was based on the layout, format and logic of the ISO BER and to a lesser extent the ISO PER documents. Therefore, the committee perceived this effort to have a minimal amount of risk as we would avoid potential problems by following the lead of other standards.
4. The design provides the bulk, if not all, of the bandwidth savings offered by PER, while providing the above benefits.

5. While it was recognized that there is always some risk in developing a new standard, it was felt that the OER risks were insignificant as compared to the other risks associated with different interpretations of object definitions. Thus, if there are minor problems with OER, they can be resolved without any significant systems impact.
6. Overall, there was a feeling that OER would require fewer processing cycles than ISO PER. However, this was seen as a minor point as (a) it has not been quantified, and (b) ISO PER may perhaps be efficient enough for most systems.

Once the NTCIP Joint Committee authorized the effort to develop OER as an NTCIP standard, it assigned the technical work to the BSP WG. The BSP WG followed the standard operating procedures of the NTCIP process. This process ensures balanced participation from both the public and private sectors to develop standards in a consensus-based process in order to meet industry needs. The document only gets elevated to a standard when there is a consensus among the group that the content defines a valid approach. The result of this cooperative effort is a set of encoding rules that are practical for our industry to implement while still meeting our bandwidth requirements.

§