

IEEE Std 1377-1997

also known as

ANSI C12.19-1997

and

Industry Canada MC-XXX

IEEE Standard for Utility Industry End Device Data Tables

Sponsor

**IEEE Standards Coordinating Committee 31
on
Automatic Meter Reading and Energy Management**

Approved 26 June 1997

IEEE Standards Board

Abstract: Functionally related utility application data elements, grouped into a single data structure for transport are described. Data may be utilized peer-to-peer or upstream to readers or billing systems by being carried by one lower layered protocol to another stack of lower layered protocol. The data structure does not change from end device to the user of the data.

Keywords: decade of tables, end device, function-limiting control (FLC) table, time-of-use (TOU) table

The Institute of Electrical and Electronics Engineers, Inc.
345 East 47th Street, New York, NY 10017-2394, USA

Copyright © 1998 by the Institute of Electrical and Electronics Engineers, Inc.
All rights reserved. Published 1998. Printed in the United States of America.

ISBN 1-55937-932-4

No part of this publication may be reproduced in any form, in an electronic retrieval system or otherwise, without the prior written permission of the publisher.

IEEE Standards documents are developed within the IEEE Societies and the Standards Coordinating Committees of the IEEE Standards Board. Members of the committees serve voluntarily and without compensation. They are not necessarily members of the Institute. The standards developed within IEEE represent a consensus of the broad expertise on the subject within the Institute as well as those activities outside of IEEE that have expressed an interest in participating in the development of the standard.

Use of an IEEE Standard is wholly voluntary. The existence of an IEEE Standard does not imply that there are no other ways to produce, test, measure, purchase, market, or provide other goods and services related to the scope of the IEEE Standard. Furthermore, the viewpoint expressed at the time a standard is approved and issued is subject to change brought about through developments in the state of the art and comments received from users of the standard. Every IEEE Standard is subjected to review at least every five years for revision or reaffirmation. When a document is more than five years old and has not been reaffirmed, it is reasonable to conclude that its contents, although still of some value, do not wholly reflect the present state of the art. Users are cautioned to check to determine that they have the latest edition of any IEEE Standard.

Comments for revision of IEEE Standards are welcome from any interested party, regardless of membership affiliation with IEEE. Suggestions for changes in documents should be in the form of a proposed change of text, together with appropriate supporting comments.

Interpretations: Occasionally questions may arise regarding the meaning of portions of standards as they relate to specific applications. When the need for interpretations is brought to the attention of IEEE, the Institute will initiate action to prepare appropriate responses. Since IEEE Standards represent a consensus of all concerned interests, it is important to ensure that any interpretation has also received the concurrence of a balance of interests. For this reason, IEEE and the members of its societies and Standards Coordinating Committees are not able to provide an instant response to interpretation requests except in those cases where the matter has previously received formal consideration.

Comments on standards and requests for interpretations should be addressed to:

Secretary, IEEE Standards Board
445 Hoes Lane
P.O. Box 1331
Piscataway, NJ 08855-1331
USA

Note: Attention is called to the possibility that implementation of this standard may require use of subject matter covered by patent rights. By publication of this standard, no position is taken with respect to the existence or validity of any patent rights in connection therewith. The IEEE shall not be responsible for identifying patents for which a license may be required by an IEEE standard or for conducting inquiries into the legal validity or scope of those patents that are brought to its attention.

Authorization to photocopy portions of any individual standard for internal or personal use is granted by the Institute of Electrical and Electronics Engineers, Inc., provided that the appropriate fee is paid to Copyright Clearance Center. To arrange for payment of licensing fee, please contact Copyright Clearance Center, Customer Service, 222 Rosewood Drive, Danvers, MA 01923 USA; (508) 750-8400. Permission to photocopy portions of any individual standard for educational classroom use can also be obtained through the Copyright Clearance Center.

Introduction

(This introduction is not part of IEEE Std 1377-1997, IEEE Standard for Utility Industry End Device Data Tables.)

The beginning of the tables was sparked by the desire of a number of people who needed a single communications protocol for their newly emerging electronics metering devices in the 1970s. One of those people was Bill Earthman of Duke Power Company, who brought a young engineer from the distribution field organization to help update the Duke Power Company mechanical metering to the new and unwieldy growing state of electronic metering. A single communications protocol was the new staff engineer's first assignment in 1980.

In 1983, a meeting was sponsored for protocols discussion. After a hotly debated day's session between memory map versus command driven techniques, engineers held further discussions and compromised on an "indirect addressing" method of manipulating and describing data that small meter processors could manage. The table concept was born.

In 1984, EEI created an adhoc committee that vigorously created a high-level language for metering, known as the "MCL" Meter Command Language. This language required too much meter microprocessor power and was rejected. Although rejected, this work provided valuable input to the continued effort with tables.

The few interested manufacturers and utilities started meeting under the auspices of EIA standard's CEBus meetings. With the help of Dick Stilwell, Kevin McDonald, Larry Schmidt, Deryl Rhoades, Michael Anderson, Ted York, Bill Beverly, John Lauletta, Brian Seal, Skeeter Wall, and the Duke Power Company engineer, the tables work became recognized as the Utility Context in the CEBus standard. The Utility Context became more than EIA wanted to be responsible for; the Tables work was orphaned and found a new home with the AMRA/IEEE SCC31 standards body navigated by Bill Rush, who urged the work forward.

The Southeastern Electric Exchange adopted the the Tables as a guideline under the Metering Committee headed by Ed Malemezian. Jean Joly of Hyro Quebec, Canada visited a number of U.S. Electric Utilities seeking a solution to the problem of proliferation of metering protocols. He found the tables work at Duke Power Company. The Canadian and U.S. collaboration started, and under the guidance of Vuong Nguyen, prodigious work was created within the Canadian "Task Force on Data Communication Protocol for Electronic Metering Devices." That task force is now under the auspices of Industry Canada. With the work of Kah-Fae Chan, Avygdor Moise, and Michel Veillette, the tables became rich with new capabilities.

ANSI C12 Subcommittee 17 became active with Electric Utilities metering communications protocol work. Under the guidance of Lynnda Ell, the table work accelerated within the ANSI C12 standards body. The C12.19 work and other works have since been carefully sheperded by Wes Ray.

All three standards bodies, ANSI C12, Industry Canada, and IEEE SCC31, decided to collectively meet to finish the tables work and establish one North American standard for metering data description. There were 12 meetings of (usually) four days. These meetings became known as "Tablefests". Bill Gibson chaired these meetings beginning with a biblical verse reminding everyone the purpose of the meeting. Terry Penn and George Stephens edited the evergrowing working document. Input from Europe was cheerfully brought through John Newbury. Thousands of hours of work by the best talent to be had produced the rich and unique fuctionality of the "Tables."

The following members participated in the working group:

Richard D. Tucker, Chair

Charlie Adkins
 Glenn Allen
 Tariq Amjed
 Michael Anderson
 Jim Andrus
 Tom Asp
 Paul Aubin
 Ron Bane
 Claire Becker-Castle
 Ludo Bertsch
 Bill Beverly
 Pierre Bezzina
 Manthi Bilinsky
 William Blair
 Donald Block
 Tom Bowling
 William Bray
 Jim Brennan
 Andrew Brock
 Tom Broe
 William Buckley
 Eddie Buckner
 Eric Buffkin
 Dennis Burman
 Martin Burns
 Randy Butler
 Jeff Buxton
 Ken Caird
 Tom Campbell
 Larry Carmichael
 Murray Carney
 Tony Centorino
 Kah-Fae Chan
 Kathy Chang
 Gerard Chevalier
 Ron Claridge
 Frances Cleveland
 Bernice Cochran
 Mike Crowley
 Robert Cruickshank
 Karen Daderko
 Mike Dalton
 Chuck Davis
 Joe Delaney
 Emil Drobber
 Bill Earthman
 Ellen Edge
 Don Eggleston
 Mark Eldeckin
 Lynnda Ell
 Paul Estes
 Ron Farquharson
 Donald Fisher
 Natan Galperin
 Stuart Garland
 Bob Garry
 Ron Genova
 Dan Gerhart
 David Gestler
 Clark Getty
 Bill Gibson
 Jeff Gill
 Jerry Gillean

Ted York, Secretary

Frank Glaser	David Propp
Greg Gomez	Harold Rabbie
Peter Gorman	Denny Radford
Dave Gorton	Wesley Ray
Bruce Gray	Richard Reed
John Grubbs	Derl Rhoades
Richard Guenther	Jeff Richardson
Michael Hajny	Steve Robinson
Paul Hargaden	Benoit Robitallie
Joseph Harley	Art Robleto
Leon Harris	Aubin Roy
Clifford Hubbard	Rick Rundus
Ivo Hug	Bill Rush
Joe Hughes	Phil Russel
Paul Hunt	Frank Safertel
Pavel Jamnik	Steve Sanislo
John Jansen	Chris Schafer
Bill Jenrette	Ben Schall
Jean Joly	George Schimmel
Clair Jones	Larry Schmidt
Lee Karsten	Richard Schmidt
John Korz	Paul Schroeder
Lawrence Kotewa	Jerome Schull
Bill Kram	Lee Schwanke
Donald Kuilmann	Felix Sciulli
Jeff Kuiper	David Scott
Michael Kunst	Howard Scott
Becky LaBrosse	Brian Seal
Trung Lang	Dick Seubert
John Lauletta	Ken Shadek
John Lennox	Tushar Shah
Mark Lockareff	Brian Simpson
Joe Lola	Joe Simpson
Ed Malemezian	Eric Smith
Brian Markwalter	Kendle Smith
John Marsh	Ivo Steklac
Kit Maughn	George Stephens
Kevin McDonald	Richard Stetler
Jay McLellan	Dick Stilwell
Al Messano	Donald Strobel
Laurel Miller	Dan Sugarman
Jon Milliken	Mike Surratt
James Mining	Victor Tamosaitis
Avygdor Moise	Paul Taylor
Andrew Morgan	Sumin Tchen
Jim Nakama	Fred Toepfer
Robert Neu	Paula Tomkinson
August Nevolo	Tim Vahlsrom
John Newbury	Jim VanderMey
Vuong Nguyen	Michael Vecchi
Jeff O'Kuiper	Michel Viellette
Dan Nordell	Kenneth Wachs
Diane Palmer	Stuart Wagonfeld
Lauren Pananen	H.A. (Sketter) Wall
Chuck Partridge	Steve Ward
Duane Paulson	Avery Washington
Jack Pazdon	Trevor White
Terry Penn	Fred Whitmore
Dan Pinney	Kurt Weichert
Andy Porten	Kent Williams
Daniel Pouliot	David Wiseman
Dale Praught	Christopher Yasko
	Gary Ziegler

The following persons were on the balloting committee:

L. A. Carmichael
Donald Fisher
Stephen Hadden
Richard Holbert
Richard Holtz
August J. Nevolo

Daniel E. Nordell
Rick Rundus
William F. Rush
George Schimmel
Jerome W. Schull
Howard Scott

Tushar K. Shah
Joseph Simpson
Michael Surratt
Richard D. Tucker
Raymond S. Turgel
Ted York

When the IEEE Standards Board approved this standard on 26 June 1997, it had the following membership:

Donald C. Loughry, *Chair*

Richard J. Holleman, *Vice Chair*

Andrew G. Salem, *Secretary*

Clyde R. Camp
Stephen L. Diamond
Harold E. Epstein
Donald C. Fleckenstein
Jay Forster*
Thomas F. Garrity
Donald N. Heirman
Jim Isaak
Ben C. Johnson

Lowell Johnson
Robert Kennelly
E. G. "Al" Kiener
Joseph L. Koepfinger*
Stephen R. Lambert
Lawrence V. McCall
L. Bruce McClung
Marco W. Migliaro

Louis-François Pau
Gerald H. Peterson
John W. Pope
Jose R. Ramos
Ronald H. Reimer
Ingo Ritsch
John S. Ryan
Chee Kiow Tan
Howard L. Wolfman

*Member Emeritus

Also included are the following nonvoting IEEE Standards Board liaisons:

Satish K. Aggarwal
Alan H. Cookson

Lisa S. Young
IEEE Standards Project Editor

Contents

1.	Scope.....	1
2.	References.....	1
3.	Definitions.....	2
3.1	Source	2
4.	General	3
4.1	Standard tables.....	3
4.2	Manufacturer tables	4
5.	Syntax	4
5.1	Descriptive syntax.....	4
5.2	Flow of information	5
5.3	Identifiers	6
5.4	Basic data types.....	6
5.5	Values and constants.....	9
5.6	Conditionals	10
5.7	Bit field	11
5.8	Set	12
5.9	Array	12
5.10	Packed record.....	13
5.11	Table	14
6.	Special data types.....	14
6.1	Character set selection	14
6.2	Non-integer formats	15
6.3	Date and time formats.....	16
6.4	Common table or procedure identifier formats.....	20
6.5	Constants.....	21
7.	Compliance	22
8.	Table transportation issues.....	23
8.1	Minimum services and parameters	23
8.2	Pending event description	26
9.	Tables.....	27
9.1	DECADE 00: Configuration tables	27
9.2	DECADE 10: Data source tables	58
9.3	DECADE 20: Register tables.....	76
9.4	DECADE 30: Local display tables	87
9.5	DECADE 40: Security tables.....	93
9.6	DECADE 50: Time-of-use tables	99
9.7	DECADE 60: Load profile tables	114
9.8	DECADE 70: History and event logs	139
9.9	DECADE 80: User defined tables	149

Annex A (normative) ID Codes for meter equipment manufacturers 156

Annex B (normative) History and event log codes 157

Annex C (normative) Default sets for decade tables..... 158

Annex D (normative) Indices for partial table access 163

IEEE Standard for Utility Industry End Device Data Tables

1. Scope

This standard defines a table structure for utility application data to be passed between an end device and a computer. It does not define device design criteria, nor specify the language or protocol used to transport that data. The purpose of the tables is to define structures for transporting data to and from end devices.

2. References

ANSI C12.10-1987: Electromechanical Watthour Meters

ECMA-94 (June 1986), 8-Bit Single Byte Coded Graphic Character Sets—Latin Alphabet No. 1 to No. 4.¹

IEEE Std 100-1996, The IEEE Standard Dictionary of Electrical and Electronics Terms.²

IEEE Std 754-1985 (Reaff 1990), IEEE Standard for Binary Floating-Point Arithmetic.

ISO/IEC 646: 1991, Information technology—ISO 7-bit coded character set for information interchange.³

ISO 7498-1: 1994, Information technology—Open Systems Interconnection—Basic reference model: The Basic Model.

ISO 8859-1: 1987, Information processing—8-bit single byte coded graphic character sets—Part 1: Latin Alphabet No. 1.

¹ECMA publications are available from the European Computer Manufacturers Association, 114 rue du Rhone, CH-1204, Geneva, Switzerland/Suisse.

²IEEE publications are available from the Institute of Electrical and Electronics Engineers, 445 Hoes Lane, P.O. Box 1331, Piscataway, NJ 08855-1331, USA.

³ISO publications are available from the ISO Central Secretariat, Case Postale 56, 1 rue de Varembé, CH-1211, Genève 20, Switzerland/Suisse. ISO publications are also available in the United States from the Sales Department, American National Standards Institute, 11 West 42nd Street, 13th Floor, New York, NY 10036, USA.

3. Definitions

For the purposes of this standard, the following definitions apply:

3.1 act: Abbreviation for ACTUAL, indicating the programmed functional capabilities of an end device.

3.2 address: Those inputs whose states select a particular cell or group of cells.

3.3 average demand: The consumption (e.g., energy, volume) recorded during the integration period divided by the integration time period.

3.4 block average demand: An average value occurring over a demand period specified by the end device (e.g., watthours/hours). The value may be saved by the end device for maximum or minimum registration.

3.5 clock: A device that generates periodic signals used for synchronization. A device that measures and indicates time. A register whose content changes at regular intervals in such a way as to measure time.

3.6 cumulative demand: An indicating demand meter in which the accumulated total of maximum demands during the preceding periods is indicated during the period after the meter has been reset and before it is reset again. NOTE—The maximum demand for any one period is equal or proportional to the difference between the accumulated readings before and after reset.

3.7 customer : The purchaser and/or user of a product or service supplied by a service provider or utility.

3.8 data encryption: The changing of the form of a data stream such that only the intended recipient can read or alter the information and detect unauthorized messages.

3.9 decade: A functional grouping of tables by application into groups of ten. The tables are numbered “X0” through “X9,” with “X” representing the decade number.

3.10 demand: The rate of consumption, e.g., power, volume/hour.

3.11 dim: Abbreviation for DIMENSION, indicating the maximum functional capability designed into an end device. *See:* **function limiting control table.**

3.12 end device: The closest device to the sensor or control point within a metering application communication system that is compliant with the utility industry end device data tables.

3.13 function-limiting control (FLC) table: The first table in each decade specifies the limits designed into an end device with respect to variables used within the decade.

3.14 load profile: The recording, storage, and analysis of consumption data over a period of time for a particular installation.

3.15 maximum demand: The highest demand measured over a selected period of time, e.g., one month.

3.16 meter: A device that measures and records the consumption or usage of the product/service.

3.17 minimum demand: The lowest demand measured over a selected period of time, such as one month.

3.18 octet: A sequence of eight bits. For application within this standard, see 5.2.

3.19 peak demand: *See:* **maximum demand.**

3.20 present average demand: An average value occurring during a current demand interval or subinterval (e.g., watts or voltamperes).

3.21 season: A calendar-specified period used for activation of rate schedules.

3.22 sliding window (rolling-interval) demand: The block demand calculated over an integration period that includes sub-intervals of previous demand calculations.

3.23 table: Functionally related utility application data elements, grouped together into a single data structure for transport.

3.24 tariff: A published list of rate schedules and terms and conditions.

3.25 time-of-use (TOU) metering: Metering equipment that separately records metered or measured quantities according to a time schedule.

3.26 utility: A provider of electricity, gas, water, telecommunications, or related services to a community.

4. General

4.1 Standard tables

Standard tables are those whose structures are specified by this standard. These should be used for both end device programming and reading. The tables provide control tables as well as data tables for a wide variety of functions to be implemented in addition to those to be defined. This standard provides for a total of 2048 standard tables, although not all are defined.

The standard tables are grouped together in decades, with each decade covering a general area of functionality. The first table of each decade, beginning with the tens decade, is referred to as a function limiting control (FLC) table. The purpose of the FLC table is to specify the designed DIMENSION (DIM) limits for variables used within the decade for the end device.

If present, the table immediately following an FLC table is called the FLC+1 table. The FLC+1 table defines the ACTUAL (ACT) limits used for variables in the end device current configuration.

If the FLC+1 table is not present, the ACTUAL (ACT) limits are identical to the designed DIMENSION (DIM) limits specified in the FLC table.

If the FLC and FLC+1 tables are not used in a decade, then the limits are defined by a default set of values. The default sets for a particular decade are defined within this standard in Annex C. Possible combinations of tables are defined in Figure 1.

	FLC table	FLC+1 table	FLC+N table	Where ACT values are found
A	Not used	Not used	Not used	None—Decade not used
B	Not used	Not used	Used	Default set, per standard
C	Not used	Used	Not used	Not practical
D	Not used	Used	Used	Decade FLC+1 table
E	Used	Not used	Not used	Not practical
F	Used	Not used	Used	Decade FLC table
G	Used	Used	Not used	Not practical
H	Used	Used	Used	Decade FLC+1 table

Figure 1—Possible combinations of FLC, FLC+1, and decade tables

4.2 Manufacturer tables

Manufacturer tables are those structures specified by individual end device vendors. They should be used to allow introduction of new innovations or to provide customer requested data structures. They also provide a path for new functions to evolve into a standard table. This standard provides access for a total of 2048 manufacturer tables.

5. Syntax

5.1 Descriptive syntax

Describing data definitions is usually accomplished within the confines of a given language and the grammar rules of that language. Since the data definitions embodied within this standard are meant to be independent of a specific language and to be capable of being implemented within the confines of any language, a method for describing the data definitions has been adopted.

Symbol	Meaning
< >	A string contained inside angle brackets is called a non-terminal. That is, while it may be viewed as a single unit it can and shall be redefined as consisting of one or more simpler elements.
:: =	This symbol is read as “is defined as.” The non-terminal that occurs on the left-hand side (LHS) of this symbol consists of the elements (non-terminals, terminals, or a combination of the two) found on the right-hand side (RHS). A line containing an LHS, :: =, and an RHS is known as a production rule.
	The vertical bar is an OR symbol. The OR symbol always occurs on the right hand side of a production where the left hand side can be defined in more than one way. The OR bar separates valid alternative right hand sides.
[]	A symbol enclosed in square brackets is optional. The production is valid whether or not it is included.
*	The superscript asterisk is known as the Kleene star. A symbol followed by the Kleene star may occur zero or more times without violating the grammar.
+	The superscript plus sign is known as the Kleene cross. A symbol followed by the Kleene cross shall occur one or more times.
{ }	Text enclosed within curly braces is a comment provided for clarity and has no impact on the syntax or data definition.

The data structure elements are indicated by **BOLD** type. Reserved words, characters, identifiers, and other symbols required for the language are also indicated by **BOLD** type.

5.2 Flow of information

The flow of information is a stream of objects of multiple “n” octet length, when “n” is greater than one. Objects are transmitted in order of their appearance in the table structure.

For purposes of transmission, the elementary data type is an octet or byte, which is eight bits in length. The bits are numbered zero to seven. Zero is the least significant bit. Figure 2 illustrates this bit numbering and ordering. The order of transmission of the bits in an octet is defined elsewhere in the appropriate OSI layer, typically the Data Link Layer (layer 2). An octet may also be referenced as a byte.

MSBit							LSBit
7	6	5	4	3	2	1	0

Figure 2—Octet bit ordering

For some data types that are more than one octet in length, it is possible to transmit them with either most or least significant byte first. The order of transmission is dictated by **DATA_ORDER**, found in the **GEN_CONFIG_TBL** (Table 00). Each data type shall indicate a specific order of transmission or refer to order of transmission in Basic Data Type Definitions (see 5.4.1). Figure 3 illustrates a multibyte structure.

MSByte							LSByte
Octet	Octet	Octet	Octet	Octet	Octet	Octet	Octet
7-0	7-0	7-0	7-0	7-0	7-0	7-0	7-0

Figure 3—Multibyte ordering

5.3 Identifiers

An identifier is a name identifying a table, a packed record, a bit field, or a member of a packed record or of a bit field.

An identifier is made of letters, and/or numbers, and/or the character ‘_’. Some suffixes have been reserved for use with specific identifiers. An identifier ending with **_TBL** identifies a table, **_RCD** identifies a packed record and **_BFLD** identifies a bit field.

The identifiers used to identify the members of a packed record or of a bit field are subdivided according to the type of data they define. These identifiers are <int-identifier>, <bool-identifier>, <set-identifier> and <array-identifier>.

<number>	:: =	<digit> ⁺
<digit>	:: =	0 to 9
<lchar>	:: =	<digit> <character>
<character>	:: =	A to Z only, upper case.
<ID>	:: =	<lchar> ⁺ [_<lchar> ⁺] [*]
<tbl-identifier>	:: =	<ID>_TBL

<rcd-identifier>	:: =	<ID>_RCD
<bflid-identifier>	:: =	<ID>_BFLD
<member-identifier>	:: =	<ID>
<int-identifier>	:: =	<member-identifier>
<bool-identifier>	:: =	<member-identifier>
<set-identifier>	:: =	<member-identifier>
<array-identifier>	:: =	<member-identifier>
<cnst-identifier>	:: =	<ID>_CNST

5.4 Basic data types

All **TYPES** shall be defined prior to use. The following basic types can be used to build more complex types:

<int>	:: =	<INT8> <INT16> <INT24> <INT32> <INT40> <INT48> <INT64>
<unsigned-int>	:: =	<UINT8> <UINT16> <UINT32>
<floating-point>	:: =	<FLOAT32> <FLOAT64>
<fill>	:: =	<FILL8> <FILL16> <FILL32>
<spec-type>	:: =	<CHAR> <NI_FMAT1> <NI_FMAT2> <LTIME_DATE> <STIME_DATE> <TIME> <DATE> <RDATE>
<base-type>	:: =	<BCD> <NIL> <int> <unsigned-int> <floating-point> <fill> <spec-type>

5.4.1 Basic data type definitions

The types are defined in Figure 4 as follows:

Type	Definition	Order of transmission
INT8	8 b signed integer, binary signed two's complement	Not applicable
INT16	16 b signed integer	DATA_ORDER in GEN_CONFIG_TBL (Table 00)
INT24	24 b signed integer	DATA_ORDER in GEN_CONFIG_TBL (Table 00)
INT32	32 b signed integer	DATA_ORDER in GEN_CONFIG_TBL (Table 00)
INT40	40 b signed integer	DATA_ORDER in GEN_CONFIG_TBL (Table 00)
INT48	48 b signed	DATA_ORDER in GEN_CONFIG_TBL (Table 00)
INT64	64 b signed integer	DATA_ORDER in GEN_CONFIG_TBL (Table 00)
UINT8	8 b unsigned integer	Not applicable
UINT16	16 b unsigned integer	DATA_ORDER in GEN_CONFIG_TBL (Table 00)
UINT32	32 b unsigned integer	DATA_ORDER in GEN_CONFIG_TBL (Table 00)
FLOAT32	Single precision floating point real number, per IEEE Std 754-1985	DATA_ORDER in GEN_CONFIG_TBL (Table 00)
FLOAT64	Double precision floating point real number, per IEEE Std 754-1985	DATA_ORDER in GEN_CONFIG_TBL (Table 00)
FILL8	8 b of zeroes, used as space holder or filler	Not applicable
FILL16	16 b of zeroes, used as space holder or filler	Not applicable
FILL32	32 b of zeroes, used as space holder or filler	Not applicable

Figure 4—Basic type definitions and order of transmission

5.4.2 Special data types

In order to add flexibility to the structure of the tables, the following data types can be used. The specific definition is determined within the tables themselves.

5.4.2.1 Signed integers

To allow signed integer values to be easily transmitted without requiring data format translations, an indicator in **GEN_CONFIG_TBL** (Table 00) is provided. This indicator, **INT_FORMAT**, specifies the data

format for all **INTx** types within the tables. The presently delineated signed integer data formats are twos complement, ones complement and sign/magnitude.

Examples of signed integer formats are as follows. In these examples, the number -1 is used.

Twos complement for **INT8**:

1111 1111

Ones complement for **INT8**:

1111 1110

Sign/magnitude for **INT8**:

1000 0001

5.4.2.2 Characters

To allow different character sets to be used within the tables, a data type of **CHAR** has been defined. **CHAR** defines both the size in octets of the character and the character set being used. Clause 6 provides a definition for **CHAR**. The value of **CHAR** is defined in the **GEN_CONFIG_TBL** (Table 00) through the use of the variable **CHAR_FORMAT**.

5.4.2.3 Non-integer numbers

To allow non-integer numeric values to be transmitted by different methods within the tables, additional data types have been defined. **NI_FMAT1** and **NI_FMAT2** are used to transmit numeric values of varying precision. The value of the variables **NI_FORMAT1** and **NI_FORMAT2**, which are found in the **GEN_CONFIG_TBL** (Table 00), define how **NI_FMAT1** and **NI_FMAT2** are to be interpreted when reading from or writing to tables in an end device. Clause 6 provides definitions for **NI_FMAT1** and **NI_FMAT2** and for the different values of **NI_FORMAT1** and **NI_FORMAT2**.

5.4.2.4 Date and time formats

To allow date and time to be transmitted by different methods within the tables, additional data types have been defined. These are **LTIME_DATE**, **STIME_DATE**, **TIME**, **DATE**, and **RDATE**. The value of the variable **TM_FORMAT**, which is found in the **GEN_CONFIG_TBL** (Table 00), defines how the various date and time formats are to be interpreted when reading from or writing to tables in an end device. Clause 6 provides definitions for the date and time formats.

5.4.3 Miscellaneous data types

The following additional types are defined:

BCD	Definition:	Each octet is two BCD values. The more significant value of each octet is in bits (7..4).
Binary bit values of	0000 to 1001	Used for the digits 0 to 9
	1010	Used for minus (–) sign
	1011	Used for blank () character
	1100	Not used
	1101	Used for decimal point (typically, a period or comma)
	1110	Not used
	1111	Not used
Order of transmission:		Not applicable

NIL Definition: Indicator for field of length zero octets.

5.5 Values and constants

5.5.1 Constants

Constants are provided for making certain tasks of table specification easier, but not for transport. An example of this is the accessing of a bit position within a set. The bit field is specified within a given table by a <constant> that is equated to a particular number.

<constant_type> ::= **CONSTANTS** <constant> **END;**

<constant> ::= <cnst-identifier> = <number>; |
<constant>⁺

The constants presently defined for the standard are found in 6.5.

5.5.2 Value

A value shall be a number, an <int-identifier> or an arithmetic operation between two values. Value is always an integer result.

<value> ::= <number> |
<cnst-identifier> |
<tbl-identifier>.<int-identifier> |
<bfl-identifier>.<int-identifier> |
-<value> |
<value> + <value> |
<value> - <value> |
<value> / <value> |
<value> * <value>

5.6 Conditionals

5.6.1 If Statements

The **IF** condition shall be a Boolean member of a bit field, or the value of a **SET**, or a relation between two values.

<condition> ::= <bfl-identifier>.<bool-identifier> |
<tbl-identifier>.<set-identifier>.(<value>) |
<value> > <value> |
<value> >= <value> |
<value> < <value> |
<value> <= <value> |
<value> ◇ <value> |
<value> = <value> |
<condition> **OR** <condition> |
<condition> **AND** <condition> |
<condition> **XOR** <condition> |
NOT <condition>

5.6.2 Case statements

A case expression identifies the member of a table that controls the case statement.

```
<case-expression>      :: =      <tbl-identifier>.<int-identifier> |
                                <bffd-identifier>.<int-identifier>
```

5.7 Bit field

Some data types do not always lend themselves to end on an octet boundary. Where these occur, they shall be logically grouped together into a bit field definition that shall end on an octet boundary. For purposes of description, the bit field is treated as the basic object for appearance in the table structure.

Figure 5 illustrates a subtype structure.

Most significant byte					Least significant byte
Octet	Octet	Octet	Octet	Octet	Octet
$((n*8)-1)..((n*8)-9)$	...	...	...	15..8	7..0

*n = # of octets used

Figure 5—Subtypes and bit field bit ordering

A bit field may contain one or more **IF** or **CASE** statements to modify its structure under some conditions. The dimension of a bit field and its octet ordering for the purpose of transport is defined in terms of a basic data type (see 5.4) as expressed in <bffd-type>.

```
<bffd-type>      :: =      TYPE <bffd-identifier> = <bffd>

<bffd>           :: =      BIT FIELD OF <unsigned-int> <bffd-member> END;

<bffd-member>    :: =      <member-identifier> : <sub-type>; |
                            <int-identifier> : INT (<value>..<>value>); |
                            <int-identifier> : UINT (<value>..<>value>); |
                            <bool-identifier> : BOOL (<value>); |
                            <bffd-case> |
                            <bffd-if> |
                            <bffd-member>+

<sub-type>       :: =      BOOL (<value>)|
                            INT (<value>..<>value>)|
                            UINT (<value>..<>value>)|
                            FILL (<value>..<>value>)

<bffd-if>        :: =      IF <condition> THEN <bffd-member> END; |
                            IF <condition> THEN <bffd-member> ELSE <bffd-member> END;

<bffd-case>      :: =      CASE <case-expression> OF <bffd-case-member> END;

<bffd-case-member> :: =      <value> : <bffd-member> |
                            <value> - <value> : <bffd-member> |
                            <bffd-case-member>+
```

5.7.1 Subtype definitions

BOOL (<value>)	Boolean operand of zero (0) for FALSE or one (1) for TRUE, in bit <i>x</i> of a bit field.
INT (<value>.. <i><value></i>)	Signed binary integer of “ <i>n</i> ” bits, represented by a range of successive bits.
UINT (<value>.. <i><value></i>)	Unsigned binary integer of “ <i>n</i> ” bits, represented by a range of successive bits.
FILL (<value>.. <i><value></i>)	Fill of “ <i>n</i> ” bits, represented by a range of successive bits. Used to fill out unused bits in an octet, or as a separator in a bit field. The value of the fill bits is always zero (0).

5.8 Set

A <set> is a collection of booleans. It is defined with its dimension in octets.

<set> :: = **SET** (<dim_octet>)

<dim_octet> :: = <value>

When evaluated for sizing of a set dimension the evaluation of <value> shall produce a non-negative integer. An example follows that illustrates the method for specifying which member of a set is to be accessed.

Example:

```

FLAGS_USED: SET (16) ;
IF FLAGS_USED.1 THEN ...
IF GEN_CONFIG_TBL.UDT_0_CONST THEN ...

```

5.9 Array

Multiple repetitions of the same data type, used to describe a single variable may be grouped together in an array. Array elements are indexed starting with zero. Arrays are always transmitted from element zero to the last element, as shown in Figure 6. The order of transmission of an element is a function of the data type being used.

Element 0	Element 1	...	Element <i>n</i> –2	Element <i>n</i> –1
-----------	-----------	-----	---------------------	---------------------

Figure 6—Single dimension array ordering

Arrays can be multidimensional. The right-most or last dimension is incremented prior to the first dimension. For a two-dimensional array, the first dimension can be thought of as a row and the second as a column. The array shall be transmitted in row order, as shown in Figures 7 and 8.

0, 0	0, 1
1, 0	1, 1

Figure 7— Two-dimension array

0, 0	0, 1	1, 0	1, 1
------	------	------	------

Figure 8—Two-dimension array ordering

An array can be of a single dimension or of multiple dimensions separated by commas. Variables shall be defined before they are used in a dimension statement.

<array> ::= **ARRAY**[<dim>] **OF** <simple-type>

<simple-type> ::= <base-type> |
 <rcd-identifier> |
 <bffd-identifier> |
 <tbl-identifier>.<rcd-identifier> |
 <tbl-identifier>.<bffd-identifier>

<dim> ::= <value> |
 <dim>[,<dim>]*

If the value of <dim> is zero then the array collapses to a zero element array and drops out of the table. When evaluated for sizing of an array the evaluation of <value> shall produce a non-negative integer.

When **CHAR** is used in an array, there is no justification implied. Full arrays are always transported. It is suggested that a blank () be used as fill or pad.

5.10 Packed record

Packed records are used to group simple types, arrays, packed records, and bit fields together. A packed record can contain one or more **IF** statements or **CASE** statements to modify its structure upon some conditions. The dimension of a packed record will vary with applied conditions.

Bit fields and packed records shall be defined before they are used in a packed record.

<rcd-type> ::= **TYPE** <rcd-identifier> = <rcd>

<rcd> ::= **PACKED RECORD** <rcd-member>

END;

<rcd-member> ::= <member-identifier> : <simple-type>; |
 <int-identifier> : <int>; |
 <int-identifier> : <unsigned-int>; |
 <set-identifier> : <set>; |
 <array-identifier> : <array>; |
 <rcd-case> |
 <rcd-if> |
 <rcd-member>+

<rcd-if> ::= **IF** (<condition>) **THEN** <rcd-member> **END;** |
 IF (<condition>) **THEN** <rcd-member> **ELSE** <rcd-member> **END;**

<rcd-case> ::= **CASE** <case-expression> **OF** <rcd-case-member> **END;**

<rcd-case-member> ::= <value> : <rcd-member> |
 <value>..<value> : <rcd-member> |
 <rcd-case-member>+

5.11 Table

Tables are used to group simple types, arrays, packed records and bit fields together into a final structure.

Bit fields and packed records shall be defined before they are used in a table.

```

<types>      :: =      <rcd-type> |
                        <bffd-type> |
                        <types>+
<table>      :: =      <types> TABLE <tbl-identifier> = <rcd-identifier>;

```

6. Special data types

In order to add flexibility to the definition of some of the data types used within the tables, special data types are provided. Their definition, once selected within the **GEN_CONFIG_TBL** (Table 00), remains consistent throughout all of the remaining used tables.

6.1 Character set selection

This selection is used to determine the structure of characters used in the tables. **CHAR_FORMAT** is the controlling selector and **CHAR** is the data type. **CHAR_FORMAT** is specified in **GEN_CONFIG_TBL**.

Value of CHAR_FORMAT	Definition of CHAR
0	Unassigned
1	ISO 7-bit coded character set for information interchange, per ISO/IEC 646: 1991. ⁴
2	ISO 8859-1: 1982 or ECMA-94 (June 1986).
3..7	Unassigned

6.2 Non-integer formats

This selection is used to determine the structure of non-integer numbers used in the tables. **NI_FORMAT1** and **NI_FORMAT2** are the controlling selectors, **NI_FMAT1** and **NI_FMAT2** provide type selection control for various data items. **NI_FORMAT1** and **NI_FORMAT2** are specified in **GEN_CONFIG_TBL**.

Value of NI_FORMAT1 and NI_FMAT2	Definition of NI_FMAT1 and NI_FMAT2
0	FLOAT64
1	FLOAT32
2	ARRAY[12] OF CHAR
3	ARRAY[6] OF CHAR
4	INT32 [Implied decimal point between fourth and fifth digits from least significant digit. (e.g., 0.0001 is represented as 1)]
5	ARRAY[6] OF BCD

⁴Information on references can be found in Clause 2.

Value of NI_FORMAT1 and NI_FMAT2	Definition of NI_FMAT1 and NI_FMAT2
6	ARRAY[4] OF BCD
7	INT24
8	INT32
9	INT40
10	INT48
11	INT64
12..15	Unassigned

6.2.1 CHAR numbers

When **CHAR** is used and **NI_FORMAT1** or **NI_FORMAT2** has a value of 2 or 3, the number shall be represented in the following way:

$\langle \text{char_number} \rangle \quad :: = \quad [_ *][+|-]\langle \text{digit} \rangle + [.\langle \text{digit} \rangle^*][\langle \text{exponent} \rangle][_ *]$
 $\langle \text{exponent} \rangle \quad :: = \quad \text{E|e|}^{\wedge}[+|-]\langle \text{digit} \rangle^+$
 $_ \quad :: = \quad \text{space character}$

A **CHAR** number representation may have any number of leading spaces; followed by an optional plus or minus sign; followed by a mandatory one or more digits; followed by an optional period and zero or more digits; followed by an optional exponent. An exponent is composed of the letter “E,” “e,” or “^”; followed by an optional plus or minus sign; followed by a mandatory one or more digits. Any number of spaces can lead or follow the $\langle \text{char_number} \rangle$ but no spaces are allowed to be embedded within the $\langle \text{char_number} \rangle$. This format ensures that clear identification of any $\langle \text{char_number} \rangle$ is possible in a stream (sequence) of $\langle \text{char_number} \rangle^+$, using the space character.

Examples:

Valid $\langle \text{char_number} \rangle$
 1.0E-7
 123.6478e+03
 1.2345
 1.^3

Invalid $\langle \text{char_number} \rangle$
 .5
 1.0 E-3
 e+03

6.3 Date and time formats

This selection is used to determine the structure of dates and times used in the tables. **TM_FORMAT** is the controlling selector and **LTIME_DATE**, **STIME_DATE**, **TIME**, **RDATE**, and **DATE** are the resulting data types.

In general, the following is true about date and time formats:

TM_FORMAT	Data type used
0	No clock in the end device
1	BCD with discrete fields for month, day, year, hour, minute, and seconds.
2	UINT8 with discrete fields for month, day, year, hour, minute, and seconds.
3	Binary value relative to a starting time as referenced by either CLOCK_TBL (Table 52) or CLOCK_STATE_TBL (Table 55):
	LTIME_DATE : 01/01/1970 @ 00:00:00, rounded to nearest second.
	STIME_DATE : 01/01/1970 @ 00:00:00, rounded to nearest minute.
	TIME : Time in seconds, since 00:00:00 local time.
4..7	Unassigned

The date and time structures are described below as packed records, but are used in the tables as data types.

6.3.1 LTIME_DATE, STIME_DATE, TIME types

```

TYPE LTIME_DATE = PACKED RECORD
  CASE GEN_CONFIG_TBL.TM_FORMAT OF
    0 :  NIL          : NIL;
    1 :  YEAR         : BCD;
        MONTH        : BCD;
        DAY          : BCD;
        HOUR         : BCD;
        MINUTE       : BCD;
        SECOND       : BCD;
    2 :  YEAR         : UINT8;
        MONTH        : UINT8;
        DAY          : UINT8;
        HOUR         : UINT8;
        MINUTE       : UINT8;
        SECOND       : UINT8;
    3 :  U_TIME       : UINT32;
        SECOND       : UINT8
  END;
END;
```

```

TYPE STIME_DATE = PACKED RECORD
  CASE GEN_CONFIG_TBL.TM_FORMAT OF
    0 :  NIL          : NIL;
    1 :  YEAR         : BCD;
        MONTH        : BCD;
        DAY          : BCD;
        HOUR         : BCD;
        MINUTE       : BCD;
    2 :  YEAR         : UINT8;
        MONTH        : UINT8;
        DAY          : UINT8;
  END;
```

```

        HOUR          : UINT8 ;
        MINUTE        : UINT8 ;
    3 :   U_TIME      : UINT32 ;
    END;
END;

TYPE TIME = PACKED RECORD
    CASE GEN_CONFIG_TBL.TM_FORMAT OF
        0 :   NIL      : NIL ;
        1 :   HOUR      : BCD ;
           MINUTE      : BCD ;
           SECOND      : BCD ;
        2 :   HOUR      : UINT8 ;
           MINUTE      : UINT8 ;
           SECOND      : UINT8 ;
        3 :   D_TIME    : UINT32 ;
    END;
END;

```

<i>Identifier</i>	<i>Value</i>	<i>Definition</i>
YEAR	00..89	Years 2000..2089
	90..99	Years 1990..1999
	> 100	Unassigned
MONTH	0	Unassigned
	1..12	Month of year
	> 12	Unassigned
DAY	0	Unassigned
	1..31	Day of month
	> 31	Unassigned
HOUR	00..23	Hour of day, 24 h basis.
	> 23	Unassigned
MINUTE	00..59	Minute of hour
	> 59	Unassigned
SECOND	00..59	Seconds of minute
	> 59	Unassigned
U_TIME	Universal time in minutes, since 01/01/1970 @ 00:00:00 as referenced by either CLOCK_TBL (Table 52) or CLOCK_STATE_TBL (Table 55).	
D_TIME	Time in seconds, since 00:00:00 local time.	

6.3.2 RDATE type

This type defines a recurrent date. This recurrence can be yearly, monthly, weekly, or based on a constant cycle of up to 64 days.

```

TYPE RDATE = BIT FIELD OF UINT16
    MONTH          : UINT (0..3) ;
    CASE MONTH OF
        1..13 :   OFFSET      : UINT (4..7) ;
    END;

```

```

                                WEEKDAY      :  UINT (8..10) ;
                                DAY           :  UINT (11..15) ;
14 :                                FILLER1   :  FILL (4..7) ;
                                WEEKDAY      :  UINT (8..10) ;
                                FILLER2      :  FILL (11..15) ;
15 :                                PERIOD     :  UINT (4..9) ;
                                DELTA        :  UINT (10..15) ;

                                END ;
END ; .

```

<i>Identifier</i>	<i>Value</i>	<i>Definition</i>
MONTH	0	Unassigned
	1..12	Month of year
	13	Action is repeated monthly
	14	Action is repeated weekly
	15	Action is repeated each PERIOD referenced by an ANCHOR_DATE plus DELTA . The ANCHOR_DATE shall be supplied separately.
OFFSET	0	No offset
	1	Advance to WEEKDAY before date entered
	2	Postpone to the first WEEKDAY on or after date entered
	3	Postpone to the second WEEKDAY on or after date entered
	4	Postpone to the third WEEKDAY on or after date entered
	5	Postpone to the fourth WEEKDAY on or after date entered
	6	Postpone to the last WEEKDAY of the MONTH on or after date entered
	7	Observe on date entered as well as day following date entered
	8	Postpone to Monday if Sunday
	9	Advance to Friday if Sunday
	10	Postpone to Monday if Saturday
	11	Advance to Friday if Saturday
	12	Postpone to Monday if Sunday or Saturday
	13	Advance to Friday if Sunday or Saturday
	14	Postpone to Monday if Sunday, advance to Friday if Saturday
	15	Do not observe date entered. Observe on day following date entered.
WEEKDAY	0..6	Sunday to Saturday
	7	Unassigned
PERIOD	0..63	0 to 63 days
DELTA	0..63	0 to 63 days

6.3.3 DATE Type

This type defines a non-recurrent date using a field having the same size as the recurrent date type “**RDATE**.”

```

TYPE DATE = BIT FIELD OF UINT16
    YEAR      :  UINT (0..6) ;
    MONTH     :  UINT (7..10) ;
    DAY        :  UINT (11..15) ;
END;

```

<i>Identifier</i>	<i>Value</i>	<i>Definition</i>
YEAR	00..89	Years 2000..2089
	90..99	Years 1990..1999
	> 100	Unassigned
MONTH	0	Unassigned
	1..12	Month of year
	> 12	Unassigned
DAY	0	Unassigned
	1-31	Day of month

6.4 Common table or procedure identifier formats

These types are provided to ease the specification of table access parameters and provide consistent specification of these table access parameters throughout the standard.

6.4.1 TABLE_IDA_BFLD Bit field

Type **TABLE_IDA_BFLD** provides the table or procedure number; a flag indicating standard or manufacturer table or procedure; a flag indicating pending status; and three additional fields for definition and use within the defining data structure.

```

TYPE TABLE_IDA_BFLD = BIT FIELD OF UINT16
    TBL_PROC_NBR      : UINT(0..10) ;
    STD_VS_MFG_FLAG   : BOOL(11) ;
    PENDING_FLAG      : BOOL(12) ;
    FLAG1              : BOOL(13) ;
    FLAG2              : BOOL(14) ;
    FLAG3              : BOOL(15) ;
END ;

```

6.4.2 TABLE_IDB_BFLD Bit field

Type **TABLE_IDB_BFLD** provides the table or procedure number; a flag indicating standard or manufacturer table or procedure; and a four bit unsigned integer for definition and use within the defining data structure.

```

TYPE TABLE_IDB_BFLD = BIT FIELD OF UINT16
    TBL_PROC_NBR      : UINT(0..10) ;
    STD_VS_MFG_FLAG   : BOOL(11) ;
    SELECTOR           : UINT(12..15) ;
END ;

```

6.4.3 TABLE_IDC_BFLD Bit field

Type **TABLE_IDC_BFLD** provides the table or procedure number; a flag indicating standard or manufacturer; a flag indicating table or procedure; and three additional fields for definition and use within the defining data structure.

```

TYPE TABLE_IDC_BFLD = BIT FIELD OF UINT16
    TBL_PROC_NBR      : UINT(0..10) ;
    STD_VS_MFG_FLAG   : BOOL(11) ;

```

```

PROC_FLAG          : BOOL(12) ;
FLAG1              : BOOL(13) ;
FLAG2              : BOOL(14) ;
FLAG3              : BOOL(15) ;
END ;

```

6.5 Constants

The following constants are defined for use in these tables. Note that these constants are not transported from device to device, but merely serve as an aid in the logical specification or selection of tables. Values for the following constants are expressed in decimal form.

TYPE CONSTANTS

```

FALSE_CNST          = 00 ;
TRUE_CNST           = 01 ;

GEN_CONFIG_TBL_CNST = 00 ;
GENERAL_MFG_ID_TBL_CNST = 01 ;
DEVICE_NAMEPLATE_TBL_CNST = 02 ;
ED_MODE_STATUS_TBL_CNST = 03 ;
PENDING_STATUS_TBL_CNST = 04 ;
DEVICE_IDENT_TBL_CNST = 05 ;
UTIL_INFO_TBL_CNST = 06 ;
PROC_INITIATE_TBL_CNST = 07 ;
PROC_RESPONSE_TBL_CNST = 08 ;

DIM_SOURCES_LIM_TBL_CNST = 10 ;
ACT_SOURCES_LIM_TBL_CNST = 11 ;
UOM_ENTRY_TBL_CNST = 12 ;
DEMAND_CONTROL_TBL_CNST = 13 ;
DATA_CONTROL_TBL_CNST = 14 ;
CONSTANTS_TBL_CNST = 15 ;
SOURCES_TBL_CNST = 16 ;

DIM_REGS_TBL_CNST = 20 ;
ACT_REGS_TBL_CNST = 21 ;
DATA_SELECTION_TBL_CNST = 22 ;
CURRENT_REG_DATA_TBL_CNST = 23 ;
PREVIOUS_SEASON_DATA_TBL_CNST = 24 ;
PREVIOUS_DEMAND_RESET_DATA_TBL_CNST = 25 ;
SELF_READ_DATA_TBL_CNST = 26 ;
PRESENT_REGISTER_SELECT_TBL_CNST = 27 ;
PRESENT_REGISTER_DATA_TBL_CNST = 28 ;

DIM_DISP_TBL_CNST = 30 ;
ACT_DISP_TBL_CNST = 31 ;
DISP_SOURCE_TBL_CNST = 32 ;
PRI_DISP_LIST_TBL_CNST = 33 ;
SEC_DISP_LIST_TBL_CNST = 34 ;

DIM_SECURITY_LIMITING_TBL_CNST = 40 ;
ACT_SECURITY_LIMITING_TBL_CNST = 41 ;
SECURITY_TBL_CNST = 42 ;

```

DEFAULT_ACCESS_CONTROL_TBL_CNST	= 43;
ACCESS_CONTROL_TBL_CNST	= 44;
KEY_TBL_CNST	= 45;
DIM_TIME_TOU_TBL_CNST	= 50;
ACT_TIME_TOU_TBL_CNST	= 51;
CLOCK_TBL_CNST	= 52;
TIME_OFFSET_TBL_CNST	= 53;
CALENDAR_TBL_CNST	= 54;
CLOCK_STATE_TBL_CNST	= 55;
TIME_REMAIN_TBL_CNST	= 56;
DIM_LP_TBL_CNST	= 60;
ACT_LP_TBL_CNST	= 61;
LP_CTRL_TBL_CNST	= 62;
LP_STATUS_TBL_CNST	= 63;
LP_DATA_SET1_TBL_CNST	= 64;
LP_DATA_SET2_TBL_CNST	= 65;
LP_DATA_SET3_TBL_CNST	= 66;
LP_DATA_SET4_TBL_CNST	= 67;
DIM_LOG_TBL_CNST	= 70;
ACT_LOG_TBL_CNST	= 71;
EVENTS_ID_TBL_CNST	= 72;
HISTORY_LOG_CTRL_TBL_CNST	= 73;
HISTORY_LOG_DATA_TBL_CNST	= 74;
EVENT_LOG_CTRL_TBL_CNST	= 75;
EVENT_LOG_DATA_TBL_CNST	= 76;
DIM_UDT_FUNC_LIM_TBL_CNST	= 80;
ACT_UDT_FUNC_LIM_CNST	= 81;
UDT_LIST_TBL_CNST	= 82;
UDT_SET_TBL_CNST	= 83;
UDT_0_TBL_CNST	= 84;
UDT_1_TBL_CNST	= 85;
UDT_2_TBL_CNST	= 86;
UDT_3_TBL_CNST	= 87;
UDT_4_TBL_CNST	= 88;
UDT_5_TBL_CNST	= 89;
END;	

7. Compliance

A device is considered to be in compliance with this standard if all of the following conditions are met:

- a) The device shall accept and act upon all services defined in Clause 8.
- b) **GEN_CONFIG_TBL** (Table 00):
 - 1) Two-way communicating devices shall transmit **GEN_CONFIG_TBL** (Table 00) on request;
 - 2) Suppliers of one-way communicating devices shall, as a minimum, supply **GEN_CONFIG_TBL** (Table 00) in some form.

- c) Any table transported between devices shall be either a standard table as defined by this standard or a manufacturer table defined by the device manufacturer. All standard and manufacturer tables supported for the current device implementation shall be specified in **GEN_CONFIG_TBL** (Table 00).
- d) Any procedure written to the device shall be either a standard procedure as defined by this standard or a manufacturer procedure defined by the device manufacturer. All standard and manufacturer procedures supported for the current device implementation shall be specified in **GEN_CONFIG_TBL** (Table 00).
- e) Some standard tables contain data structures that can be collapsed by a control flag. The control flag shall be determinable based on this standard. Collapsed fields shall not be transported. An array of zero elements is a collapsed field.
- f) A device is not required to use every data structure within a table; however, it shall transport all non-collapsed data structures within a table. Unused fields are considered in compliance if byte count and format are correct and if manufacturers identify unused fields. It is recommended that unused fields be filled with zeros or spaces.
- g) This standard specifies variable formats for some data structures. The variable formats shall be consistent with the format as specified in **GEN_CONFIG_TBL** (Table 00). Refer to Clauses 5 and 6 for details.
- h) Dimensional function limiting tables (FLC) and actual function limiting tables (FLC+1) specify the designed and usable limits for variables used within a given decade. (For a complete explanation of function limiting tables, see 4.1.) The permissible combinations are listed in Figure 1.

8. Table transportation issues

The effective transport of table structures is dependent only on the presence of basic read and write services. While the marketplace may dictate the incorporation of more sophisticated services, it is left to the implementors of specific protocol stacks to select the specific read and write services to be included.

8.1 Minimum services and parameters

8.1.1 Read service

This service causes the transfer of data from the initiating device to the target device and is required for two-way communications.

Read request	The read request shall be capable of both a complete and a partial table read. This is accomplished by one required parameter and optional parameters. The required parameter is a table identifier. Optional parameters define partial table retrieval.
Table identifier	This parameter has a range of 0–8191. This range is further subdivided as follows: a) The range from 0–2047 provides access to standard tables 0–2047 b) The range from 2048–4095 provides access to manufacturer tables 0– 2047. c) The range 4096–6143 provides access to standard pending tables 0–2047. d) The range 6144–8191 provides access to manufacturer pending tables 0– 2047.
Partial table retrieval	The retrieval of a portion of a table, a partial table, is possible through the use of one of two methods. These methods are offset-count and index-count. The absence of either of these methods implies the entire table shall be read.

Offset count method	The offset-count method is based upon supplying an offset from the beginning of the selected table and, optionally, a count of the number of octets to read from the offset point. The range of the offset from the beginning of the table is 0 to a minimum of 65 535 octets. The range of the count of the number of octets to read from the offset point is 0 to a minimum of 65 535 octets. A zero for the count value or the absence of the count value shall cause will imply to read the entire remaining table to be read starting atfrom the offset point.
Indexed-count method	The index-count method is based upon accommodating a minimum of 5 indices relative to the first data structure within the table and, optionally a count of the number of octets to read from the index point. Each index shall range from 0 to a maximum of 65 535 octets. The range of the count of the number of octets to read from the index point is 0 to a minimum of 65 535 octets. A zero for the count value or the absence of the count value shall cause to read the entire remaining table to be read starting at the index point.
Read response data	The read response shall contain as a minimum the table data selected by the aforementioned parameters supplied with the read request. When the read request table identifier specifies a pending table, a six-octet pending event description precedes the requested information. The pending event description parameters are described in 8.2.

8.1.2 Write service

This service causes the transfer of unrequested data to a target device from the initiating device and is required for both one- and two-way communications.

Write request	The write request shall be capable of both a complete and a partial table write. This is accomplished by one required parameter and partial table identifiers and the write data field. The required parameter is a table identifier. Optional parameters define partial table write.
Table identifier	a) This parameter has a range of 0–8191. This range is further subdivided as follows: b) The range from 0–2047 provides access to standard tables 0–2047. c) The range from 2048–4095 provides access to manufacturer tables 0–2047. d) The range 4096–6143 provides access to standard pending tables 0–2047. Several additional parameters are required for writing to a pending table. These parameters are appended to the front of the table data to be written and are described in the Table Data Write section. e) The range 6144–8191 provides access to manufacturer pending tables 0–2047. Several additional parameters are required for writing to a pending table. These parameters are appended to the front of the table data to be written as a pending event description and are described in 8.2.
Partial table identifier	The writing of a portion of a table, a partial table, is possible through the use of one of two methods. These methods are offset-count and index-count. The absence of either of these methods implies the entire table shall be written.

Offset-count method	The offset-count method is based upon supplying an offset from the beginning of the selected table and, optionally, a count of the number of octets to write to the offset point. The range of the offset from the beginning of the table is 0 to a minimum of 65 535 octets. The range of the count of the number of octets to write to the offset point is 0 to a minimum of 65 535 octets. A zero for the count value, or the absence of the count value, shall cause to write the entire remaining table to be written from the offset point.
Indexed-count method	The indexed method is based upon accommodating a minimum of 5 indices relative to the first data structure within the table and, optionally, a count of the number of octets to write to the index point. Each index shall range from 0 to a maximum of 65 535 octets. The range of the count of the number of octets to write shall range from 0 to a maximum of 65 535 octets. A zero for the count value or the absence of the count value shall imply writing the entire remaining table from the index point.
Data	The data field contains the information to write. When the write request table identifier specifies a pending table, a six-octet pending event description precedes the information in the data field.
Write response	As a minimum, no response is required to the write request. This does not preclude use of write responses supported by implementor languages or their supporting protocols.

8.2 Pending event description

Data read from or written into a pending table is identical to data read from or written into non-pending tables in structure, with the exception of the addition of a preceding pending event descriptor.

```

TYPE  STATUS_BFLD  = BIT FIELD OF UINT8
      EVENT_CODE    :  UINT(0..3) ;
      SELF_READ_FLAG:    :  BOOL(4) ;
      DEMAND_RESET_FLAG: :  BOOL(5) ;
      RESERVED      :  FILL(6..7) ;
END ;

TYPE  PENDING_EVENT_DESC_RCD = PACKED RECORD
      EVENTS_SELECTOR:    :  STATUS_BFLD ;
      EVENT_STORAGE:      :  ARRAY[5] OF UINT8 ;
END ;

```

<i>Identifier</i>	<i>Value</i>	<i>Definition</i>
STATUS_BFLD		Selectors indicating action to be performed upon activation of selected table.

EVENT_CODE		Variable that specifies the condition upon which the pending table shall be made active. The trigger condition is stored in the EVENT_STORAGE variable.
	0	The pending table is activated based on an absolute time contained in the EVENT_STORAGE variable.
	1	The pending table is activated based on a relative time contained in the EVENT_STORAGE variable.
	2	The pending table is activated based on a non-time associated trigger contained in the EVENT_STORAGE variable.
	3..15	Reserved.
SELF_READ_FLAG	FALSE	Do not perform a self read before pending table is activated.
	TRUE	Perform a self read, if capable, before pending table is activated.
DEMAND_RESET_FLAG	FALSE	Do not perform a demand reset before pending table is activated.
	TRUE	Perform a demand reset, if capable, before pending table is activated.
RESERVED		Reserved for future use.
PENDING_EVENT_DESC_RCD		
EVENTS_SELECTOR		Status bits indicating various conditions associated with a pending table.
EVENT_STORAGE		Variable for storage of the trigger condition that causes a pending table to become active.
STIME_DATE	0	For an EVENT_CODE of 0 this variable contains the year, month, day, hour, and minute in the format indicated.
ARRAY[5] OF UINT8	1	For an EVENT_CODE of 1 this variable contains an offset time where the array positions indicate the following: [0] weeks; [1] days; [2] hours; [3] minutes; and [4] seconds.
CHAR	2	For an EVENT_CODE of 2 this variable contains in the first four characters the ASCII manufacturer code per Annex A of this standard. The final character is an event number as defined by the manufacturer.

9. Tables

9.1 DECADE 00: Configuration tables

This decade contains tables associated with end device configuration, identification, procedures and responses, and information required for data manipulation due to hardware configurations.

9.1.1 TABLE 00 General configuration table**Table 00 Data Description**

GEN_CONFIG_TBL (Table 00) contains general end device configuration information. It also establishes the total set of tables, procedures, and the selection of special types used in the end device. If a default set is used, it is indicated by this table.

```

TYPE FORMAT_CONTROL_1_BFLD = BIT FIELD OF UINT8
    DATA_ORDER          : UINT(0..0);
    CHAR_FORMAT          : UINT(1..3);
    MODEL_SELECT         : UINT(4..6);
    FILLER               : FILL(7..7);
END;

TYPE FORMAT_CONTROL_2_BFLD = BIT FIELD OF UINT8
    TM_FORMAT            : UINT(0..2);
    DATA_ACCESS_METHOD  : UINT(3..4);
    ID_FORM              : UINT(5..5);
    INT_FORMAT           : UINT(6..7);
END;

TYPE FORMAT_CONTROL_3_BFLD = BIT FIELD OF UINT8
    NI_FORMAT1          : UINT(0..3);
    NI_FORMAT2          : UINT(4..7);
END;

TYPE GEN_CONFIG_RCD = PACKED RECORD
    FORMAT_CONTROL_1      : FORMAT_CONTROL_1_BFLD;
    FORMAT_CONTROL_2      : FORMAT_CONTROL_2_BFLD;
    FORMAT_CONTROL_3      : FORMAT_CONTROL_3_BFLD;
    MANUFACTURER          : ARRAY[4] OF CHAR;
    NAMEPLATE_TYPE        : UINT8;
    DEFAULT_SET_USED       : UINT8;
    MAX_PROC_PARM_LENGTH  : UINT8;
    MAX_RESP_DATA_LEN     : UINT8;
    STD_VERSION_NO        : UINT8;
    STD_REVISION_NO       : UINT8;
    DIM_STD_TBLS_USED      : UINT8;
    DIM_MFG_TBLS_USED      : UINT8;
    DIM_STD_PROC_USED      : UINT8;
    DIM_MFG_PROC_USED      : UINT8;
    DIM_MFG_STATUS_USED    : UINT8;
    NBR_PENDING           : UINT8;
    STD_TBLS_USED          : SET(GEN_CONFIG_TBL.DIM_STD_TBLS_USED);
    MFG_TBLS_USED          : SET(GEN_CONFIG_TBL.DIM_MFG_TBLS_USED);
    STD_PROC_USED          : SET(GEN_CONFIG_TBL.DIM_STD_PROC_USED);
    MFG_PROC_USED          : SET(GEN_CONFIG_TBL.DIM_MFG_PROC_USED);
    STD_TBLS_WRITE        : SET(GEN_CONFIG_TBL.DIM_STD_TBLS_USED);
    MFG_TBLS_WRITE        : SET(GEN_CONFIG_TBL.DIM_MFG_TBLS_USED);
END;

TABLE GEN_CONFIG_TBL = GEN_CONFIG_RCD;

```

<i>Identifier</i>	<i>Value</i>	<i>Definition</i>
FORMAT_CONTROL_1_BFLD		Variable to define order of multibyte numeric data transfer.
DATA_ORDER	0	Least significant byte first.
	1	Most significant byte first.
CHAR_FORMAT		See Character Set Selection in Special Data Types, 6.1. Unsigned binary code where:
	0	Unassigned.
	1	ISO 7-bit coded character set for information interchange, per ISO/IEC 646: 1991.
	2	ISO 8859-1: 1987 or ECMA-94 (June 1986).
	3..7	Unassigned.
MODEL_SELECT		Unsigned binary code where:
	0	A code indicating the I/O model selects sources as an 8 b index into the DATA_CONTROL_TBL (Table 14).
	1..7	Reserved
FORMAT_CONTROL_2_BFLD TM_FORMAT		Data type used for dates and time:
	0	No clock in the end device
	1	BCD with discrete fields for month, day, year, hour, minute and seconds.
	2	UINT8 with discrete fields for month, day, year, hour, minute and seconds.
	3	UINT32 with time referenced as seconds relative to a starting time. The reference point is as follows: LTIME_DATE : 01/01/1970 @ 00:00:00 Greenwich Mean Time (GMT), rounded to nearest second. STIME_DATE : 01/01/1970 @ 00:00:00 GMT, rounded to nearest minute. TIME : Time in seconds since 00:00:00 local time (including time zone offset).
	4..7	Unassigned
DATA_ACCESS_METHOD		Variable to designate method of passing the table data after GEN_CONFIG_TBL (Table 00) has been interrogated. NOTE—All data access methods support complete table access.
	0	Complete table access only. No partial table access supported.
	1	Offset-count access method is supported.
	2	Index-count method is supported.
	3	Both methods 1 and 2 of data access method are supported.
ID_FORM		Indicates the format of the identifier field in DEVICE_IDENT_TBL (Table 05).

	0	CHAR string.
	1	BCD string.
INT_FORMAT		Indicates the format of signed integer types. Signed integer types are represented in twos complement format. Signed integer types are represented in ones complement format. Signed integer types are represented in sign/magnitude format. Reserved.
FORMAT_CONTROL_3_BFLD		
NI_FORMAT1		Indicates the type of non-integer value used throughout the tables where specified as NI_FMAT1 . Refer to 6.2 for details.
NI_FORMAT2		Indicates the type of non-integer value throughout the tables where shown as NI_FMAT2 . Refer to 6.2 for details.
GEN_CONFIG_RCD		
FORMAT_CONTROL_1		See descriptions above.
FORMAT_CONTROL_2		See descriptions above.
FORMAT_CONTROL_3		See descriptions above.
MANUFACTURER		end device manufacturer identification. Refer to Annex A for list of manufacturer identifiers.
NAMEPLATE_TYPE		Entry used to select the nameplate record structure to be used in DEVICE_NAMEPLATE_TBL (Table 02).
	0	Gas
	1	Water
	2	Electric
	3..255	Unassigned
DEFAULT_SET_USED		Indicates which, if any, default sets are used. See Annex C for the default set definitions.
	0	No default values in use. See 4.1, Figure 1, conditions C, D, E, and F for more detail.
	1	Default set #1, Simple Meter Register, in use.
	2	Default set #2, Simple Demand Meter, in use.
	3	Default set #3, Simple TOU Meter, in use.
	4	Default set #4, Simple Profile Recorder, in use.
	5..255	Reserved for default table value sets.
MAX_PROC_PARM_LENGTH		Manufacturer defined maximum length for argument parameters passed to procedures in PROC_INITIATE_TBL (Table 07).
MAX_RESP_DATA_LEN		Manufacturer defined maximum length representing length returned by procedures in PROC_RESPONSE_TBL (Table 08).

STD_VERSION_NO		Unsigned binary number designating the version of the particular industry set of standard tables.
	0	Pre-release or informal release document.
	1	Original standard table set, first release document.
	2..255	Reserved for Standards Committee use only.
STD_REVISION_NO		Unsigned binary number that designates a minor change to a version of this standard. Within a given version of this standard, all revisions with a lower revision number shall be backward compatible. If this is not true then a new version number shall be required.
DIM_STD_TBLS_USED		Octets required to represent the set of standard tables used in GEN_CONFIG_TBL.STD_TBLS_USED (Table 00). This is a mechanism to allow the table identifier set to expand in size from simple devices to complex devices.
DIM_MFG_TBLS_USED		Octets required to represent the set of manufacturer tables used in GEN_CONFIG_TBL.MFG_TBLS_USED (Table 00). This is a mechanism to allow the table identifier set to expand in size from simple devices to complex devices.
DIM_STD_PROC_USED		Octets required to represent the set of standard procedures used in GEN_CONFIG_TBL.STD_PROC_USED (Table 00). This is a mechanism to allow the standard defined procedure identifier set to expand in size.
DIM_MFG_PROC_USED		Octets required to represent the set of manufacturer procedures used in GEN_CONFIG_TBL.MFG_PROC_USED (Table 00). This is a mechanism to allow the manufacturer defined procedure identifier set to expand in size.
DIM_MFG_STATUS_USED	0..255	The number of octets allocated for indicating manufacturer specific status flags.
NBR_PENDING		Number of pending status sets in PENDING_STATUS_TBL (Table 04).
STD_TBLS_USED		This set variable indicates which of the standard tables are implemented by the device. Tables are represented by bits 0 through $(8 \times \text{DIM_STD_TBLS_USED} - 1)$, with a 1 representing a TRUE or implemented condition and a 0 representing a FALSE or not implemented condition.

MFG_TBLS_USED

This set variable indicates which of the manufacturer tables are implemented by the device. Tables are represented by bits 0 through $(8 * \text{DIM_MFG_TBLS_USED} - 1)$, with a 1 representing a TRUE or implemented condition and a 0 representing a FALSE or not implemented condition.

STD_PROC_USED

This set variable indicates which of the standard procedures are implemented by the device. Procedures are represented by bits 0 through $(8 * \text{DIM_STD_PROC_USED} - 1)$, with a 1 representing a TRUE or implemented condition and a 0 representing a FALSE or not implemented condition.

MFG_PROC_USED

This set variable indicates which of the manufacturer's procedures are implemented by the device. Procedures are represented by bits 0 through $(8 * \text{DIM_MFG_PROC_USED} - 1)$, with a 1 representing a TRUE or implemented condition and a 0 representing a FALSE or not implemented condition.

STD_TBLS_WRITE

This set variable indicates which of the standard tables are capable of being rewritten. Tables are represented by bits 0 through $(8 * \text{DIM_STD_TBLS_USED} - 1)$, with a 1 representing a writeable table and a zero representing a table that cannot be written.

MFG_TBLS_WRITE

This set variable indicates which of the standard tables are capable of being rewritten. Tables are represented by bits 0 through $(8 * \text{DIM_MFG_TBLS_USED} - 1)$, with a 1 representing a writeable table and a zero representing a table that cannot be written.

9.1.2 TABLE 01 General manufacturer identification table**Table 01 Data description**

GENERAL_MFG_ID_TBL (Table 01) is a general manufacturer identification data table for water, gas, and electric utility end devices. The values of the data items shall be set up by the manufacturer and left unchanged since they are determined by the functionality of the device.

TYPE MANUFACTURER_IDENT_RCD = PACKED RECORD

```

MANUFACTURER          : ARRAY[4] OF CHAR;
ED_MODEL              : ARRAY[8] OF CHAR;
HW_VERSION_NUMBER     : UINT8;
HW_REVISION_NUMBER    : UINT8;
FW_VERSION_NUMBER     : UINT8;
FW_REVISION_NUMBER    : UINT8;
IF GEN_CONFIG_TBL.ID_FORM THEN

```

```

        MFG_SERIAL_NUMBER : ARRAY[8] OF BCD;
    ELSE
        MFG_SERIAL_NUMBER : ARRAY[16] OF CHAR;
    END;
END;

```

```
TABLE GENERAL_MFG_ID_TBL = MANUFACTURER_IDENT_RCD;
```

<i>Identifier</i>	<i>Value</i>	<i>Definition</i>
MANUFACTURER_IDENT_RCD		
MANUFACTURER		Name of manufacturer. Refer to Annex A.
ED_MODEL		Model identifier of the end device left justified. For example, 'ENG M3.'
HW_VERSION_NUMBER	0..255	Manufacturer's hardware version number. Implies functional changes.
HW_REVISION_NUMBER	0..255	Manufacturer's hardware revision number. Implies product corrections or improvements. Recommend use of "0" to designate prototyping device.
FW_VERSION_NUMBER	0..255	Manufacturer's firmware version number. Changes in value imply functional differences.
FW_REVISION_NUMBER	0..255	Manufacturer's firmware revision number. Changes in value imply product corrections or improvements. Recommend use of "0" to designate preproduction or prototyping software.
MFG_SERIAL_NUMBER		Manufacturer's serial number for the end device. 16 BCD digits OR CHARs .

9.1.3 TABLE 02 Device nameplate table

Table 02 Data description

DEVICE_NAMEPLATE_TBL (Table 02) is the nameplate data table. This table contains a record structure for each type of end device (e.g., water, gas, electric). The structure to be used is specified in **GEN_CONFIG_TBL.NAMEPLATE_TYPE** (Table 00). This table presently has nameplate structures defined for an electric meter, a gas meter, and a water meter.

{water device structures}

```

TYPE W_WATER_DEVICE_BFLD = BIT FIELD OF UINT16
    W_ED_TYPE           : UINT(0..2);
    W_FLUID_TYPE        : UINT(3..6);
    W_ED_DRIVE          : UINT(7..10);
    W_ED_PIPE_SIZE      : UINT(11..15);
END;

```

{gas device structures}

```

TYPE G_ED_TYPE_BFLD = BIT FIELD OF UINT8
    G_ED_TYPE           : UINT(0..2);
    G_MECH_FORM         : UINT(3..5);

```

```

        G_ENG_METRIC      : UINT(6..6);
        FILLER            : FILL(7..7);
END;

TYPE G_PRESSURE_RCD = PACKED RECORD
    G_MAX_PRESS          : NI_FMAT2;
    G_UOM_PRESS          : UINT8;
END;

TYPE G_FLOW_RCD = PACKED RECORD
    G_MAX_FLOW           : NI_FMAT2;
    G_UOM_FLOW           : UINT8;
END;

TYPE G_SIZE_BFLD = BIT FIELD OF UINT8
    G_GEAR_DRIVE         : UINT(0..2);
    G_INPUT_OUTPUT_PIPE  : UINT(3..7);
END;

TYPE G_COMPENSATION_BFLD = BIT FIELD OF UINT8
    G_COMP_TEMP          : UINT(0..4);
    G_COMP_PRESS         : UINT(5..7);
END;

TYPE G_GAS_DEVICE_RCD = PACKED RECORD
    G_ED_TYPE            : G_ED_TYPE_BFLD;
    G_MAX_PRESS          : G_PRESSURE_RCD;
    G_MAX_FLOW           : G_FLOW_RCD;
    G_GEAR_PIPE_SIZE     : G_SIZE_BFLD;
    G_COMPENSATION       : G_COMPENSATION_BFLD;
END;

{electric device structures}
TYPE E_ELEMENTS_BFLD = BIT FIELD OF UINT16
    E_FREQ              : UINT(0..2);
    E_NO_OF_ELEMENTS    : UINT(3..5);
    E_BASE_TYPE         : UINT(6..9);
    E_ACCURACY_CLASS     : UINT(10..15);
END;

TYPE E_VOLTS_BFLD = BIT FIELD OF UINT8
    E_ELEMENTS_VOLTS     : UINT(0..3);
    E_ED_SUPPLY_VOLTS    : UINT(4..7);
END;

TYPE E_AMPS_RCD = PACKED RECORD
    E_CLASS_MAX_AMPS     : ARRAY[6] OF CHAR NI_FMAT2;
    E_TA                 : ARRAY [6] OF CHAR NI_FMAT2;
END;

TYPE E_ELECTRIC_DEVICE_RCD = PACKED RECORD
    E_kh                 : ARRAY [6] OF CHAR NI_FMAT2;
    E_kt                 : ARRAY [6] OF CHAR NI_FMAT1;
    E_INPUT_SCALAR       : UINT8;
    E_ED_CONFIG          : ARRAY[5] OF CHAR;

```

```

        E_ELEMENTS                : E_ELEMENTS_BFLD;
        E_VOLTS                   : E_VOLTS_BFLD;
        E_AMPS                    : E_AMPS_RCD;
    END;

    TYPE DEVICE_DEFINITION_RCD = PACKED RECORD
        CASE GEN_CONFIG_TBL.NAME_PLATE_TYPE OF
            0      : G_GAS_DEVICE           :G_GAS_DEVICE_RCD;
            1      : W_WATER_DEVICE         :W_WATER_DEVICE_BFLD;
            2      : E_ELECTRIC_DEVICE      :E_ELECTRIC_DEVICE_RCD;
            3..255 : RESERVED                :NIL;
        END;
    END;

    TABLE DEVICE_NAMEPLATE_TBL = DEVICE_DEFINITION_RCD;

```

The following structures are generally associated with water utility end devices. The values of the data items shall be set up by the manufacturer and left unchanged since they are determined by the functionality of the device.

<i>Identifier</i>	<i>Value</i>	<i>Definition</i>
W_WATER_DEVICE_BFLD		
W_ED_TYPE		Code representing water end device type.
	0	Meter
	1	Valve
	2	Pressure Regulator
	3	Pressure gauge
	4	Backflow
	5	Other
	6-7	Unassigned
W_FLUID_TYPE		Code representing type of fluid measured by water end device.
	0	Potable water
	1	Hot water
	2	Non-potable water
	3	Sewage primary water
	4	Sewage secondary water
	5	Sewage tertiary water
	6	Other
	7	Unassigned
W_ED_DRIVE		Code representing water end device drive.
	0	Piston
	1	Disc
	2	Multijet
	3	Turbine
	4	Compound
	5	Propeller
	6	Ultrasonic
	7	Magnetic-coupled
	8	Differential pressure
	9	Mass

10	Variable area
11	Open channel
12	Oscillatory
13	Other
14..15	Unassigned

W_ED_PIPE_SIZE

Code representing end device pipe size.

0	1/2 in	13 mm
1	5/8 in	15 mm
2	3/4 in	20 mm
3	1 in	25 mm
4	1 1/2 in	40 mm
5	2 in	50 mm
6	3 in	80 mm
7	4 in	100 mm
8	6 in	160 mm
9	8 in	200 mm
10	10 in	250 mm
11	12 in	300 mm
12	14 in	350 mm
13	16 in	400 mm
14	18 in	450 mm
15	20 in	500 mm
16	22 in	Not standard
17	24 in	600 mm
18	26 in	Not standard
19	28 in	Not standard
20	30 in	Not standard
21	32 in	800 mm
22	34 in	Not standard
23	36 in	900 mm
24	40 in	1000 mm
25	48 in	1200 mm
26..31	Unassigned	

The following structures are generally associated with gas utility end devices. The values of the data items shall be set up by the manufacturer and left unchanged since they are determined by the functionality of the device.

<i>Identifier</i>	<i>Value</i>	<i>Definition</i>
-------------------	--------------	-------------------

G_ED_TYPE_BFLD
G_ED_TYPE

Code representing gas end device type.

0	Unclassified
1	Gas meter
2	Gas pressure regulator
3	Sensor
4	Load shed device
5	Cathodic protection unit
6..7	Unassigned

G_MECH_FORM		Unsigned binary number identifying the primary mechanical design principle of the device depending upon value of G_ED_TYPE .
	For G_ED_TYPE = 1.	
	0	Unclassified
	1	Bellows meter
	2	Rotary
	3	Turbine meter
	4	Fluidic oscillator
	5	Anemometer
	6..7	Unassigned
	For G_ED_TYPE = 2.	
	0	Unclassified
	1	Diaphragm
	2..7	Unassigned
	For G_ED_TYPE = 3.	
	0	Unclassified
	1	Physical parameter to voltage
	2	Physical parameter to current
	3	Physical parameter to frequency
	4	Open/closed switch
	5..7	Unassigned
	For G_ED_TYPE = 5.	
	0	Unclassified
	1	Pipe-to-soil
	2	Impedance
	3..7	Unassigned
G_ENG_METRIC	0	English
	1	Metric
G_PRESSURE_RCD		
G_MAX_PRESS		End device maximum design pressure.
G_UOM_PRESS		See field ID_CODE in UOM_ENTRY_TBL (Table 12).
G_FLOW_RCD		
G_MAX_FLOW		End device maximum design flow rate.
G_UOM_FLOW		See field ID_CODE in UOM_ENTRY_TBL (Table 12).
G_SIZE_BFLD		
G_GEAR_DRIVE		Code representing gas meter gear drive size.
	0	None
	1	1/2 ft

2	1 ft
3	2 ft
4	5 ft
5	10 ft
6	100 ft
7	1000 ft

G_INPUT_OUTPUT_PIPE

Code representing gas meter pipe size.

	English	Metric
0	None	None
1	1/2 in	
2	5/8 in	
3	3/4 in	
4	1 in	
5	1 1/2 in	
6	2 in	
7	4 in	
8	6 in	
9	8 in	
10	10 in	
11	12 in	
12	14 in	
13	16 in	
14	18 in	
15	20 in	
16	22 in	
17	24 in	
18	26 in	
19	28 in	
20	30 in	
21	32 in	
22	34 in	
23	36 in	
24..31	Unassigned	

G_COMPENSATION_BFLD**G_COMP_TEMP**

Code representing the gas meter temperature compensation type.

0	Uncompensated
1	Mechanical
2	Sensor
3..7	Unassigned

G_COMP_PRESS

Code representing the gas meter pressure compensation type.

0	Uncompensated
1	Mechanical
2	Sensor
3..7	Unassigned

The following structures are generally associated with electric utility end devices. The values of the data items shall be set up by the manufacturer and left unchanged since they are determined by the functionality of the device.

<i>Identifier</i>	<i>Value</i>	<i>Definition</i>
E_ELEMENTS_BFLD		Contains binary codes physically describing the end device.
E_FREQ		Power frequency rating code. The codes are:
	0	DC
	1	25 Hz
	2	50
	3	50 or 60 Hz
	4	60 Hz
	5	400 Hz
	6	Unassigned
	7	Unclassified
E_NO_OF_ELEMENTS		Number of commodity measuring elements per measuring input to the end device. This code indicates the number of elements as follows:
	0	None
	1	1 element
	2	2 elements
	3	2.5 elements
	4	3 elements
	5	6 elements
	6	1.5 elements
	7	Unassigned
E_BASE_TYPE		Indicates the type of meter base as follows:
	0	None
	1	S-base (socket)
	2	A-base (ANSI bottom-connected)
	3	K-base
	4	IEC bottom-connected
	5	switchboard
	6	rackmount
	7	B-base
	8	P-base (Canadian Standard)
	9–15	Unassigned
E_ACCURACY_CLASS		Reserved for future ANSI solid-state meter standard accuracy class definitions.
E_VOLTS_BFLD		Contains binary codes describing meter RMS voltages.
E_ELEMENTS_VOLTS		Meter element voltage code. This binary code indicates the meter voltage class as follows:
	0	None
	1	69.3
	2	72
	3	120
	4	208
	5	240

	6	277	
	7	480	
	8	120–277	
	9	120–480	
	10–15	Unassigned	
E_ED_SUPPLY_VOLTS			External supply voltage code. This value identifies the meter supply voltage as follows:
	0	Internal	
	1	69.3 ac	
	2	72 ac	
	3	120 ac	
	4	208 ac	
	5	240 ac	
	6	277 ac	
	7	480 ac	
	8	120–277 ac	
	9	120–480 ac	
	10	48 dc	
	11	125 dc	
	12	250 dc	
	13–15	Unassigned	
E_AMPS_RCD			
E_CLASS_MAX_AMPS			End device class or IEC max amp rating.
E_TA			The RMS amperage test amps (TA) specified by the manufacturer for the main test and/or adjustment of the meter.
E_ELECTRIC_DEVICE_RCD			
E_kh			Watt-hours per revolution. (May be represented as unit-hours per equivalent revolution).
E_kt			The commodity amount selected for the test pulse output.
E_INPUT_SCALAR			Divisor by which to scale input values. For example if input consists of pulses, an INPUT_SCALAR value of 2 would cause the pulse stream to be divided by 2.
E_ED_CONFIG			Form number per ANSI C12.10.
DEVICE_DEFINITION_RCD			This is full device nameplate data structure configured as a case statement for selection of water, gas or electric end devices. The selection variable is GEN_CONFIG_TBL.NAMEPLATE_TYPE (Table 00).

9.1.4 TABLE 03 ED_MODE Status table

Table 03 Data description

ED_MODE_STATUS_TBL (Table 03) provides the current operating mode and status of various conditions in the end device. It allows for both standard and manufacturer specific status definitions.

```

TYPE ED_MODE_BFLD = BIT FIELD OF UINT8
    METERING_FLAG          : BOOL(0);
    TEST_MODE_FLAG         : BOOL(1);
    METER_SHOP_MODE_FLAG   : BOOL(2);
    FILLER                  : FILL(3..15);
END;

TYPE ED_STD_STATUS1_BFLD = BIT FIELD OF UINT16
    UNPROGRAMMED_FLAG      : BOOL(0);
    CONFIGURATION_ERROR_FLAG : BOOL(1);
    SELF_CHK_ERROR_FLAG    : BOOL(2);
    RAM_FAILURE_FLAG       : BOOL(3);
    ROM_FAILURE_FLAG       : BOOL(4);
    NONVOL_MEM_FAILURE_FLAG : BOOL(5);
    CLOCK_ERROR_FLAG       : BOOL(6);
    MEASUREMENT_ERROR_FLAG : BOOL(7);
    LOW_BATTERY_FLAG       : BOOL(8);
    LOW_LOSS_POTENTIAL_FLAG : BOOL(9);
    DEMAND_OVERLOAD_FLAG   : BOOL(10);
    POWER_FAILURE_FLAG     : BOOL(11);
    FILLER                  : FILL(12..15);
END;

TYPE ED_STD_STATUS2_BFLD = BIT FIELD OF UINT8
    FILLER                  : FILL(0..7);
END;

TYPE ED_MFG_STATUS_RCD = PACKED RECORD
    ED_MFG_STATUS          : SET(GEN_CONFIG_TBL.DIM_MFG_STATUS_USED);
END;

TYPE ED_MODE_STATUS_RCD = PACKED RECORD
    ED_MODE                : ED_MODE_BFLD;
    ED_STD_STATUS1         : ED_STD_STATUS1_BFLD;
    ED_STD_STATUS2         : ED_STD_STATUS2_BFLD;
    ED_MFG_STATUS          : ED_MFG_STATUS_RCD;
END;

TABLE ED_MODE_STATUS_TBL = ED_MODE_STATUS_RCD;

```

<i>Identifier</i>	<i>Value</i>	<i>Definition</i>
TYPE ED_MODE_BFLD METERING_FLAG	FALSE	End device is not in metering mode. It may or may not be accumulating measured or input quantities.
	TRUE	End device is in metering mode and is accumulating measured or input quantities.
TEST_MODE_FLAG	FALSE	End device is not in test mode.
	TRUE	End device is in test mode.

METER_SHOP_MODE_FLAG	FALSE	End device is not in meter shop mode.
	TRUE	End device is in meter shop mode.
TYPE ED_STD_STATUS1_BFLD		
UNPROGRAMMED_FLAG	FALSE	End device is programmed.
	TRUE	End device is not programmed or it is in a factory default state.
CONFIGURATION_ERROR_FLAG	FALSE	End device did not detect a configuration error.
	TRUE	End device did detect a configuration error.
SELF_CHK_ERROR_FLAG	FALSE	End device did not detect a self check error.
	TRUE	End device did detect a self check error.
RAM_FAILURE_FLAG	FALSE	End device did not detect a RAM memory failure.
	TRUE	End device did detect a RAM memory failure.
ROM_FAILURE_FLAG	FALSE	End device did not detect a ROM memory failure.
	TRUE	End device did detect a ROM memory failure.
NONVOL_MEM_FAILURE_FLAG	FALSE	End device did not detect a nonvolatile memory failure.
	TRUE	End device did detect a nonvolatile memory failure.
CLOCK_ERROR_FLAG	FALSE	End device did not detect a clock error.
	TRUE	End device did detect a clock error.
MEASUREMENT_ERROR_FLAG	FALSE	End device did not detect a measurement element error.
	TRUE	End device did detect a measurement element error.
LOW_BATTERY_FLAG	FALSE	End device did not detect a low-battery error.
	TRUE	End device did detect a low-battery error.
LOW_LOSS_POTENTIAL_FLAG	FALSE	End device did not detect a potential that is below a predetermined value.
	TRUE	End device did detect a device potential that is below a predetermined value.

DEMAND_OVERLOAD_FLAG	FALSE	End device did not detect a demand threshold overload.
	TRUE	End device did detect a demand threshold overload.
POWER_FAILURE_FLAG	FALSE	End device did not detect a power failure.
	TRUE	End device did detect a power failure.
ED_STD_STATUS2_BFLD		Standard status code bit field 2 is a place holder for future expansion.
ED_MFG_STATUS_RCD		Set containing the manufacturer specific status flags.

ED_MODE_STATUS_RCD	
ED_MODE	See ED_MODE_BFLD .
ED_STD_STATUS1	See ED_STD_STATUS1_BFLD .
ED_STD_STATUS2	See ED_STD_STATUS2_BFLD .
ED_MFG_STATUS	See ED_MFG_STATUS_RCD .

ED_MODE**9.1.5 TABLE 04 Pending status table****Table 04 Data description**

PENDING_STATUS_TBL (Table 04) is established to provide an indication of the tables present in the end device with a pending status. Information contained within this table includes table number, manufacturer or standard table indicator, conditions (events) upon which activation of a pending table is to occur, self-read or demand reset upon activation indicators, and several additional status variables.

```

TYPE STATUS_BFLD = BIT FIELD OF UINT8
    EVENT_CODE           : UINT(0..3);
    SELF_READ_FLAG       : BOOL(4);
    DEMAND_RESET_FLAG    : BOOL(5);
    RESERVED              : FILL(6..7);
END;

TYPE EVENT_RCD = PACKED RECORD
    EVENT_SELECTOR       : STATUS_BFLD;
    EVENT_STORAGE        : ARRAY[5] OF UINT8;
END;

TYPE ENTRY_ACTIVATION_RCD = PACKED RECORD
    EVENT                : EVENT_RCD;
    TABLE_SELECTOR      : TABLE_IDA_BFLD;
END;

TYPE PENDING_ACTIVATION_RCD = PACKED RECORD
    STANDARD_PENDING     : SET(GEN_CONFIG_TBL.DIM_STD_TBLS_USED);
    MANUFACT_PENDING     : SET(GEN_CONFIG_TBL.DIM_MFG_TBLS_USED);
    LAST_ACTIVATION_DATE_TIME : STIME_DATE;
    NBR_PENDING_ACTIVATION : UINT8;
    PENDING_TABLES       : ARRAY[GEN_CONFIG_TBL.NBR_PENDING] OF
        ENTRY_ACTIVATION_RCD;
END;

```

TABLE PENDING_STATUS_TBL = PENDING_ACTIVATION_RCD;

Identifier	Value	Definition
STATUS_BFLD		Selectors indicating action to be performed upon activation of selected table.
EVENT_CODE	0	Variable that specifies the condition upon which the pending table shall be made active. The trigger condition is stored in the EVENT_STORAGE variable. The pending table is activated based on an absolute time contained in the EVENT_STORAGE variable.

	1	The pending table is activated based on a relative time contained in the EVENT_STORAGE variable.
	2	The pending table is activated based on a non-time associated trigger contained in the EVENT_STORAGE variable.
	3..15	Reserved.
SELF_READ_FLAG	FALSE	Do not perform a self read before pending table is activated.
	TRUE	Perform a self read, if capable, before pending table is activated.
DEMAND_RESET_FLAG	FALSE	Do not perform a demand reset before pending table is activated.
	TRUE	Perform a demand reset, if capable, before pending table is activated.
RESERVED		Reserved for future use.
ENTRY_ACTIVATION_RCD EVENTS_SELECTOR		Status bits indicating various conditions associated with a pending table.
EVENT_STORAGE		Variable for storage of the trigger condition that causes a pending table to become active.
STIME_DATE		For an EVENT_CODE of 0, this variable contains the year, month, day, hour, and minute in the format indicated.
STIME_DATE		For an EVENT_CODE of 1, this variable contains the weeks, days, hours, minutes, and seconds in the format indicated.
CHAR		For an EVENT_CODE OF 2 this variable contains in the first four characters the ASCII manufacturer code per Annex A of this standard. The final character is an event number as defined by the manufacturer.
TABLE_SELECTOR		Variable containing the table number, the standard or manufacturer flag and the pending flag. The pending flag indicates that the associated table has not been activated.
PENDING_ACTIVATION_RCD STANDARD_PENDING		This set variable indicates which of the standard tables are capable of being written with a pending status.
MANUFACT_PENDING		This set variable indicates which of the manufacturer tables are capable of being written with a pending status.
LAST_ACTIVATION_DATE_TIME		Date and time of the last pending table activated.