Registrar Certification Testing Requirements for IEEE 1377/ANSI C12.19/MC12.19 TDL

Version 1.0

(Revised March 30, 2012)

Prepared by

Future DOS R&D Inc. and KEMA, Inc. (“Authors”)

Approved by NAEDRA
March 30, 2012

NOTICE AND DISCLAIMER The following is a description of requirements for testing. The language assumes that the reader has complete fluency in the languages of IEEE 1377, ANSI C12.19 and MC12.19 Standards including their respective definition sections, semantics and architectures. The definitions and terms used in the Standards may not correspond directly to the every-day or “IT” language. These terms have IEEE 1377 / ANSI C12.19 / MC12.19 domain specific meanings. The reader is strongly advised to read the definition section of the Standards or, preferably, consult an expert who has first-hand knowledge of AMI and Smart Grid implementations that are based on the these Standards.

The ANSI Standards were published in 1997 and 2008. The IEEE 1377 Standards were published in 1997 (and balloted in 2010). After the publication of ANSI C12.19 in 2008 a number of errors were discovered and corrections were applied. These errors and corrections were identified and applied in 2011 by the corresponding IEEE, ANSI and MC (Measurement Canada) documents, consistent with the signed memorandum of understanding among ANSI (NEMA), IEEE (SCC31) and MC. The authors, as set forth above, assume no responsibility and do not accept any liability for any content, errors or omissions that may exist in this document, whether patent or latent. Also the authors make no warranties or guarantees and hold no responsibility, what-so-ever, for any damage, injury or loss of property that may arise from the use of this document and shall be held harmless for same. The authors shall not be responsible for any guarantee of compliance or failed compliance with the NAEDRA Requirements and Testing Procedures as set forth herein, or any other applicable body.

Terminology The keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 Ref. [8].

Revision History

Date	Revision	Description	Author
2010-11-02	0.1	Initial revision. Principles introduces for KEMA/NAEDRA valuation	FDOS
2010-11-12	0.2	Editorial corrections applied.	FDOS/KEMA
2011-01-04	0.3	New section 4.0 was created.	FDOS/KEMA
2011-01-07	0.4	Corrected typographic errors and capitalization.	FDOS/KEMA
2011-03-07	0.5	Added new TERMINOLOGY section. Added new references in Section 6. Updated sections 2,3 and 4.	FDOS/KEMA
2011-03-23	0.6	Added new TERMINOLOGY section. Added new references in Section 6. Updated sections 2, 3 and 4.	FDOS/KEMA
2011-06-24	0.7	NAEDRA corrections applied per Greenville meeting. Editorial changes applied to sections 2.3, 4.2, 4.8 and 6.	NAEDRA
2011-10-04	0.8	Editorial changes. Inserted Revision History, spacing and corrected references and English. Converted to docx format.	FDOS
2011-10-11	0.9	Completed section 5.	FDOS/KEMA
2011-12-15	0.10	Changes title and document name and applied minor editorial corrections to examples in section 5.	FDOS/KEMA
2012-01-04	1.0 draft	Accepted all the changes that were made in draft 0.10. Submitted to NAEDRA for approval as Version 1.0.	FDOS
2012-03-15	1.0 draft	Began working on the feedback document submitted by Brent Cain.	NAEDRA
2012-03-23	1.0 draft	Continued work on the feedback document submitted by Brent Cain.	NAEDRA
2012-03-30	Version 1.0	Continued work on the feedback document submitted by Brent Cain. Document was moved from DRAFT to v1.0	NAEDRA

Contents

List of Figures

List of Tables

1 Introduction

This document defines requirements for testing a submission of a TDL/XML Form representation (Ref. [1] and Ref. [2], Annex I.1.1 TDL/EDL Files and Terminology) of the ANSI C12.19-2008, Ref. [1], as corrected by IEEE P1377-2011, Ref. [2]. For brevity we collectively refer to Ref. [1] and Ref. [2] as "ANSI C12.19" or "the ANSI C12.19 Standard", whether referred in singular or plural.

The ANSI C12.19 Standard binds a C12.19 Device Class to an End Device Operating Model (Ref. [1] and Ref. [2], Annex L.8 Binding a Device Class to End Device Operating Model). The C12.19 Standard semantics of the Data Model are defined by the standard in two forms:

1. The Document Form (Ref. [1] and Ref. [2], Annex G: Document-form Descriptive Syntax); and
2. The XML Form (Ref. [1] and Ref. [2], Annex I: XML File Format of TDL and EDL Files).

The Document Form is used to publish the Standard in a human readable and comprehensible form. The XML Form is used to publish the same information in a machine readable form. The TDL/XML Form SHALL be used by conforming Head-end systems and meter readers. These are collectively described by ANSI C12.19 as the "AMI Application" (Ref. [1] and Ref. [2], Figure I.5 – From XML to AMI application. The pathways for using C12.19 Standard and Manufacturer-defined TDL/XML tables for Documentation…). For this reason, the Standard is written in a manner that makes it possible to generate the Document Form from the XML Form using strict rules defined by the Standard.

The production requirements of the Document Form are found in Ref. [1] and Ref. [2], Section 5, Syntax; and are fully defined in Annex G, "Document-form Descriptive Syntax. The ANSI C12.19 Standard was published in accordance with this syntax. The syntax of the production rules of the XML Form and their rendering to Document Form are described in Ref. [1] and Ref. [2], Annex I: XML File Format of TDL and EDL Files. Also, Ref. [1] and Ref. [2], Annex I.1.1 TDL/EDL Files and Terminology, identifies a process that can be used to convert C12.19 Standard XML Form to ANSI C12.19 Standard Document Form (Ref. Figure I.1 – Production of the Document Form (Document Format of Section 9.0, "Tables") from the TDL XML File). The XML Form document is referred to in the ANSI C12.19 Standard as the "TDL" or "Table Definition Language" (Ref. [1] and Ref. [2], Section 3.81 Table Definition Language (TDL)).

The XML Form can be used to describe the ANSI C12.19 standard data types, Tables syntax, Procedures syntax and their meta-data using XML notations that may appear complex to the untrained reader. The C12.19 Standard Document Form (as published) is designed to be read by persons. Therefore, in order to affirm the correctness of a Registrar's candidate TDL/XML Form (the "CANDIDATE") it is necessary to compare the CANDIDATE's TDL submission against the published standard Document Form. The accreditation of the CANDIDATE's submission is performed by an accreditation agency (the "ACCREDITOR").

2 CANDIDATE TDL/XML Form Accreditation Process

A simple testing / verification process is described in this document (Ref. Proposal to NAEDRA for a process for testing Registrar TDLs Ref. [7]). This process can be used to certify that the TDL/XML-Form File that was produced by the CANDIDATE is accurate in accordance with reference to the ANSI C12.19 Standard. As such an assertion SHALL be made by the CANDIDATE stating that it's TDL/XML Form, when certified, is an accurate and equivalent representation of the referenced ANSI C12.19 standard's data types, Tables syntax, Procedures syntax and their meta-data (the "SEMANTICS") and in accordance with the production requirement stipulated in Annex G and I of the applicable ANSI C12.19 Standards.

2.1 Basic Principles

1. The ACCREDITOR is engaged in a manner that is similar to an activity of a Notary Public. The process is exemplified below:

    a. A Notary Public accepts two documents from a client. One document is an original and the other is a translation or a copy of the original; then;

    b. the Notary Public compares the two documents for identity (or substantial equivalence); finally

    c. the Notary Public certifies that the "copy" of the "original" document is an accurate rendering of the original document.

    The ACCREDITOR does not need to be an expert in the document's subject matter, but it needs to have a capability to independently confirm the equivalence in form and content of the two documents.

    The "original", in the above example, is the ANSI C12.19 standard(s) Document Form that was referenced by the CANDIDATE. The "copy" is the machine rendered Document Form (presented as a PDF, DOC or any printable form) of the relevant Document Form sections of the referenced Standard (these sections are described later on in this document). The standard sections "copy" shall be rendered from a TDL/XML-Form that was created by the CANDIDATE, who claims it to be an accurate transcription of the referenced ANSI C12.19 standard's SEMANTICS.

2. The ACCREDITOR deploys a simple process (See Section 2.3, The DOCUMENT Test Procedure) to convert the CANDIDATE's submitted TDL/XML Form file into a Document-Form (the "DOCUMENT"), that is expected to be a correct representation of the Standard's SEMANTICS. Therefore, this is the DOCUMENT that SHALL be used by the ACCREDITOR in the comparison and verification process against the Standard Document Form's SEMANTICS.

3. In order to gain confidence in the impartiality of the process. The ACCREDITOR may apply changes to the CANDIDATE's TDL/XML Form file. These changes shall be applied in a manner that is not known by the CANDIDATE so that the ACCREDITOR can be satisfied that the machine rendered DOCUMENT manifests the alterations that were introduced by the ACCREDITOR, therefore, rendering the CANDIDATE's conversion process trust-worthy. This way the ACCREDITOR is not burdened by complexity of designing and commissioning the DOCUMENT production process; and the CANDIDATE is not constrained by the ACCREDITOR's ability to generate a DOCUMENT.

4. In order to gain additional confidence in the quality of the supplied material, the ACCREDITOR MAY inspect the CANDIDATE's submitted TDL/XML Form file. This SHOULD be used to determine conformance to XML encoding rules, in accordance with XML-2008 Ref. [9], and TDL/XML Form construction rules in accordance with Annex I, XML File Format of TDL and EDL Files, of Ref. [1] and Ref. [2].

These principles form core requirement 1 for the accreditation of the CANDIDATE's TDL/XML Form file.

2.2 The DOCUMENT Production Principles

A well-produced TDL/XML Form contains data types, data structure, elements, enumerations, assertions, associations, qualifications, properties, scopes, controls and documentation that

SHALL be complete and encoded according to SEMANTICs in Annex I and the constraints in Annex G (Ref. [1] and Ref. [2]). When the TDL/XML Form is rendered into a DOCUMENT the result SHOULD be comparable to the Document Form of Sections 4, 6, 8 and 9 of ANSI C12.19 (Ref. [1] and Ref. [2]) and Annex C of ANSI C12.22 (Ref. [3] and Ref. [4]).

These production principles form core requirement 2 for the accreditation of the CANDIDATE's TDL/XML Form file.

2.3 The DOCUMENT Test Procedure

The process described herein assumes that the CANDIDATE will provide the ACCREDITOR access to a WEB tool for the purpose of conversion of the TDL/XML Form to DOCUMENT. CANDIDATES may choose to offer other mechanisms for rendering their TDL/XML Form to DOCUMENT, at their option. The process that is described below minimizes the impact on the ACCREDITOR, while meeting core requirements 1 and 2 that were identified the previous sections.

The detailed acceptance/rejection criteria for the DOCUMENT are discussed fully later on in this document.

The process is as follows:

1. The ACCREDITOR and the CANDIDATE sign mutual non-disclosure and non-distribution agreements to protect their respective intellectual property.
2. The CANDIDATE formally submits to the ACCREDITOR an electronic file representation of the CANDIDATE's TDL/XML Form file (e.g. on a CD). This file shall be used by the ACCREDITOR in its validation and certification process.
3. The ACCREDITOR inspects the submitted TDL/XML File so that it is satisfied that it is indeed an XML document and it is well-formed according to the Extensible Markup Language (XML) rules for encoding documents in machine-readable form. These rules are defined in the XML 1.0 Specification Ref. [9]. There are many commercially available tools and public domain available tools the can perform this task. NAEDRA SHOULD assist the ACCREDITOR, by providing a list of independent tools that can be used to test whether the XML file is well-formed.
4. The CANDIDATE provides temporary access to the ACCREDITOR (e.g. via its web site) to the CANDIDATE's TDL/XML Form to DOCUMENT conversion tool.
5. The ACCREDITOR executes the tests using the following general sequence:
    a. The ACCREDITOR uploads the CANDIDATE TDL/XML Form electronically to the CANDIDATE's web-site.
    b. The ACCREDITOR invokes the TDL/XML Form to DOCUMENT conversion tool (e.g. by click of a button).
    c. The ACCREDITOR downloads the generated DOCUMENT (e.g. in PDF format) from the CANDIDATE's website.
    d. The ACCREDITOR performs the necessary comparison between the CANDIDATE's DOCUMENT and the Standard's Document Form SEMANTICS.
6. Step (5) above SHALL be repeated by the ACCREDITOR, at least once, with injected (valid) variations to the TDL/XML-Form file. This enables the ACCREDITOR to see how the variations manifest themselves in the resulting DOCUMENT. This step can also provide the ACCREDITOR an opportunity to inspect the TDL/XML File. For more helpful information and examples on this subject See Ref. [2] Annex I.2.2.15 <description> Document Form Annex I.2.4.3 <enumerator> Document Form (named), Annex I.2.8.3

<packedRecord> Document Form and similar section in Annex I of Ref. [1] and Ref. [2] that contain the words “Document Form” in their title.

If the ACCREDITOR is satisfied with the results of the comparison then the source TDL/XML Form file that was submitted by the CANDIDATE SHALL be approved. Otherwise, if there are significant discrepancies that cannot be resolved by the CANDIDATE, the submission will be rejected.

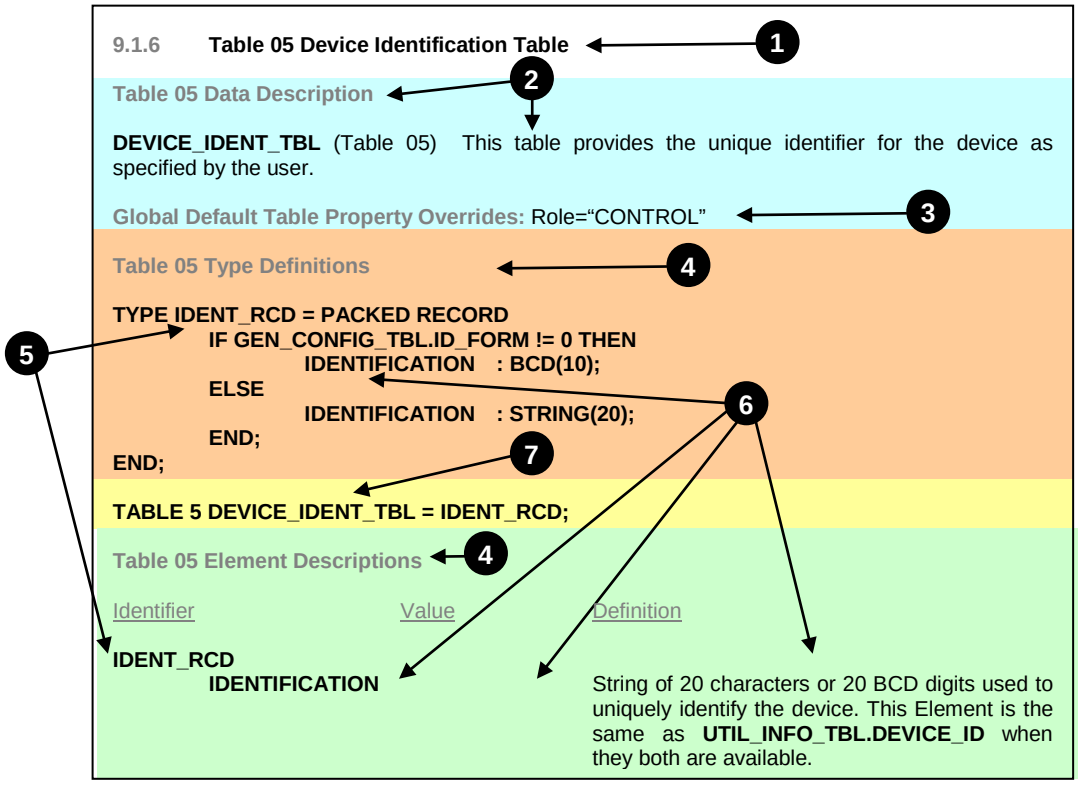
Upon completion of the tests, the ACCREDITOR SHALL return all of the submitted documents, files and tools to the CANDIDATE then it SHALL destroy all copies that may be left over on the ACCREDITOR computers and archival systems.

3 Document Comparison Overview

3.1 Document Form Elements of One Table

Figure 1 shows the ANSI C12.19 published Standard Table 05. This table is very simple. Some of the key required elements that are published by ANSI C12.19 Standard Table are as shown below.

Figure 1: Sample DOCUMENT or Standard Document Form Table Definition.



In the above example identified are some of the published elements from ANSI C12.19 (Ref. [1] and Ref. [2], Section 9.1.6 Table 05 Device Identification Table). The requirements for publishing those elements are fully defined by the ANSI C12.19 Standard.

Figure 1 shows Table elements as presented by the C12.19 Standard. Ideally Figure 1 should be identical to rendered DOCUMENT. Figure 1 is color coded. The color represents DOCUMENT regions that need to be produced from CANDIDATE’s TDL/XML Form file.

Following the “style” of ANSI C12.19, for publishing Tables, these regions are:

1. Table introducer section (in white) where the Table's Section Heading is introduced. In this section the Table number (e.g. "5") is bound to a "graphic" label that describes the table (e.g. "Device Identification Table").
2. Table description (in sky-blue), where the Table is described in plain text English and its qualifying attributes are listed (e.g. this Table has the role of a control table that contains End Device identification information).
3. Table syntax definitions (in orange). This section annotates in pseudo-Pascal language the data structures that are defined in the scope of this Table. These may be expressed using PACKED RECORD, BIT FIELDS or enumerators. Each data structure is a container for data element definitions and condition statements (e.g. IF/THEN/ELSE or SWITCH/CASE).
4. The instantiation section (in yellow). A C12.19 Table and its related data structure cannot be accessed unless a table is instantiated (defined). A C12.19 Table is created by the TABLE directive. This directive strictly binds a Table name (e.g. "DEVICE_IDENT_TBL") to a Table number (e.g. "5", which is also bound to the "graphic" label) and to a PACKED RECORD structure (e.g. "IDENT_RCD").
5. Table syntax description (in light green). This is where each of the Table's defined or used data types and their elements are described in plain English. Additionally, each data type and their elements may be qualified using certain attributes (e.g. volatile= "false", none are shown in the Figure 1) that further qualify the characteristics of the data type or its elements).

3.2 Required Elements in a Document Form of a Table Definition

Rendering a DOCUMENT from a TDL / XML Form may produce varying content. The variation of the content of a DOCUMENT is minimized by rules published in ANSI C12.19. The color in code Figure 1 also includes grayed text areas, that represent a "style" that is used by the Standard, but it is not required. Table 1 identifies the DOCUMENT production rules, thus the acceptance requirements for comparing Tables in a DOCUMENT against Tables in the Standard. Elements that are marked with an "O" are optional. Elements that are marked with an "M" are mandatory as a production rule for the DOCUMENT. Mandatory elements represent substantive content that MUST be available to all users of the TDL/XML Form. As a result they MUST be manifested accurately in the DOCUMENT in support of an interoperable AMI interface. Note: The black circled numbers in Table 1 refer to the black circled numbers in Figure 1

Table 1: Requirements for the production of TDL/Document Form from TDL/XML Form

Document Form Region	Region Description	M/O	ANSI C12.19 Reference
❶ Table introducer	Introduces a Table section definition which includes a table name, a number and a label. The **name, number, label and alias** are published in the Table section heading of the Table. Their values SHALL be exactly as written in the Standard. Caveats: 1. The ANSI C12.19 Standard uses leading zeros as in "Table 00" when expressing numeric values. The use of leading zeros is not required.	M	Annex I.2.24 <table> element, Annex I.2.28 <table> element

	Document Form Region	Region Description	M/O	ANSI C12.19 Reference
2	Table data description	Describes the C12.19 Table. This is formatted text that SHALL match the wording of the Standard. **Caveats**: 1. The text format of the words may not be identical to that found in the Standard.	M	I.2.2 <description> element
3	Table property override	Describes the non-default C12.19 Table properties names and values. The complete list of properties is: **role**, **associate**, **atomic**, **accessibility**, **deprecated**, **metrological**, **volatile** and **class**. The above properties SHALL be present in the Table's description section. Minimally non-default properties SHALL be listed. Their value SHALL be identical to the value published by the Standard. **Caveats**: 1. If the properties names are capitalized (for esthetic reason, as done by the Standard, e.g. Role="CONTROL") then the TDL/XML Form should be checked to ensure that all property attribute names found <table> attributes are entered in lower case letters, as prescribed in Section I.2.24.1, <table> DTD of the ANSI Standard.	M	Section 4.1.2 Standard Tables Properties, Annex I.2.24 <table> element Annex I.2.28 <table> element
4	Table Syntax Types Definitions Introducer	This text acts to separate the C12.19 Table description section from the Table's types and elements syntax.	O	Although it is shown in the Standard it is not documented as a required heading.

5	Table Types Syntax	This is an introducer of C12.19 data type wrappers. Possible types of wrappers are:
		`TYPE XXX_RCD = PACKED RECORD ….`
		`… elements`
		`END;`
		
		`TYPE YYY_BFLD = BIT FIELD OF`
		`… sub-elements`
		`UINTx … END;` and
		
		`{ Enumerator ZZZ_ENUM }`
		
		The syntax is case sensitive and all keywords SHALL comply with the definitions found in Annex G of ANSI C12.19. These SHALL be entered exactly as written in the Standard for each type defined, using the published **name** and **type**.
		
		If the type is defined elsewhere and then redefined here in the Table's scope then the referenced type's wrapper and definition SHOULD be omitted from this Table Types Syntax section.
			M	ANNEX G: Document-form Descriptive Syntax,
				Annex I.2.8 <packedRecord> Element,
				Annex I.2.15 <bitField> element,
				I.2.4 <enumerator> Element

	Document Form Region	Region Description	M/O	ANSI C12.19 Reference
		Caveats: 1. Font style and indentation and white space may differ from the Standard. 2. Given that the Standard was typed in by hand and the DOCUMENT is produced by machine it is conceivable that the Standard may contain syntax error. These errors MAY be documented and explained by the CANDIDATE and SHALL be considered by the ACCREDITOR.		

6	Table Elements Syntax	This is a collection of a data type's internal element declaration and conditional statements. These are defined inside PACKED RECORDs or BIT FIELD containers that are directly or indirectly referenced by the PACKED RECORD that is equated with the Table.
		e.g.
		`TYPE IDENT_RCD = PACKED RECORD`
		`  IDENTIFICATION: STRING(20);`
		`END;`
		
		All elements and conditional syntax SHALL be entered exactly as written in the Standard. The syntax is case sensitive and all keywords SHALL comply with the definitions found in Annex G of ANSI C12.19.
		
		Caveats:
		1. Font style and indentation and white space may differ from the Standard.
		2. Given that the Standard was typed in by hand and the DOCUMENT is produced by machine it is conceivable that the Standard may contain syntax error. These errors MAY be documented and explained by the CANDIDATE and SHALL be considered by the ACCREDITOR.
			M	ANNEX G: Document-form Descriptive Syntax, Annex I.2.8, <packedRecord> Element,
				Annex I.2.9 <element> Element through Annex I.2.22 <default>

7	Table Instance Definition Syntax	This is where the C12.19 Table is defined. e.g. `TABLE 5 DEVICE_IDENT_TBL=IDENT_RCD;` The Table's **name, number, and type** attributes are published in the Table's definition statement that follows the Table's type definitions. The syntax SHALL be entered exactly as written in the Standard. The syntax is case sensitive and all keywords SHALL comply with the definitions found in Annex G of ANSI C12.19. **Caveats**:	M	Annex G.13 Tables, Annex I.2.24 <table> Element

	Document Form Region	Region Description	M/O	ANSI C12.19 Reference
		1. Font style and indentation and white space may differ from the Standard. 2. Given that the Standard was typed in by hand and the DOCUMENT is produced by machine it is conceivable that the Standard may contain syntax error. These errors MAY be documented and explained by the CANDIDATE and SHALL be considered by the ACCREDITOR. 3. The ANSI C12.19 Standard uses leading zeros as in "Table 00" when expressing numeric values. The use of leading zeros is not required.		
4	Table types and elements descriptions Introducer	This text acts to separate the types and elements syntax from its elements descriptions.	O	Although it is shown in the Standard it is not documented as a required heading.
5	Table Types Description	Each of the data types listed in the Table Types Syntax SHALL be described in type's description section. The description SHALL include the type's **name** and type's description. If the type is defined elsewhere and then redefined in this Table's scope type's property names and values SHALL be listed for **replace**, **override**, **redefine**. Minimally the non-default type's properties names and values SHALL be listed for **strict**, **alias**, **atomic**, **accessibility**, **deprecated** and **metrological** properties. The properties values value SHALL be identical to the published value in the Standard. Named enumerated types SHALL be described **Caveats**: 1. The text format of the type description may not be identical to that found in the Standard. 2. If the properties names are capitalized (for esthetic reason, as done by the Standard, e.g. Atomic="true") then the TDL/XML Form should be checked to ensure that all property attribute names found for the type's (e.g. <packedRecord>, <bitField>) attributes are entered in lower case letters, as prescribed as prescribed by ANSI C12.19 for the attributes.	M	Annex I.2.8 <packedRecord> Element, Annex I.2.15 <bitField> element, I.2.4 <enumerator> Element Annex I.2.31 <packedRecord> Element Annex I.2.32 <bitField>
6	Table Elements, description,	Each of the data types listed in (5), Table Types Syntax, SHALL be included in type's description section. The description SHALL include the type's	M	Annex I.2.8 <packedRecord> Element,

Document Form Region	Region Description	M/O	ANSI C12.19 Reference
enumerations and assertions Description	**name** and type's description. If the type is defined elsewhere and then redefined in the Table's scope type's property names and values SHALL be listed for **replace**, **override**, **redefine**. Minimally the non-default element and sub-element property names and values SHALL be listed for **alias**, **atomic**, **accessibility**, **deprecated, metrological and volatile**. The properties values value SHALL be identical to the published value. If the element or sub-element is enumerated or is an enumerator then the enumerated items (value/description pairs) SHALL be listed as after the elements, sub-element or named enumerator's description. Enumerated constant names SHALL be presented in front of the enumerator's description (between value and description). For un-enumerated element or sub-element then properties **min**, **max** SHALL express the range of values allowed by the element or sub-element, exactly as indicated by the Standard (using '..' as range delimiters). If assertion is indicated then an **assert** description including its **condition**, **raise** and **text** values SHALL be present (for an example of what assets looks like See Annex L.2.1.1 Case 1: Exposing Manufacturer's Content Elaboration). **Caveats**: 1. The text format of the type description may not be identical to that found in the Standard. 2. If the properties names are capitalized (for esthetic reason, as done by the Standard, e.g. Atomic="true") then the TDL/XML Form should be checked to ensure that all property attribute names found for the type's (e.g. <element>, <subElement>, <array>, <set>, <binary>, <string>) attributes are entered in lower case letters, as prescribed as prescribed by ANSI C12.19 for these attributes.		Annex I.2.15 <bitField> element, I.2.3 <assert> element I.2.4 <enumerator> Element Annex I.2.31 <packedRecord> Element Annex I.2.32 <bitField> I.2.6.2 <enum> Attributes **I.2.30 <element> Element (Child of <qualify>)**

3.3 Example test DOCUMENTs

3.3.1 Test DOCUMENT is Valid

The following document should be compared with Figure 1: Sample DOCUMENT or Standard Document Form Table Definition.

Figure 2: Sample Valid DOCUMENT.

```
Table 5 Device Identification Table

DEVICE_IDENT_TBL (Table 05)
This table provides the unique identifier for the device as specified by the user.

Non default properties: role="CONTROL"

Table 05 Type Definitions

TYPE IDENT_RCD = PACKED RECORD IF GEN_CONFIG_TBL.ID_FORM != 0 THEN
IDENTIFICATION    : BCD(10); ELSE
IDENTIFICATION                                          : STRING(20);
        END;END;

TABLE 5 DEVICE_IDENT_TBL = IDENT_RCD;

Descriptions for Table 5

IDENT_RCD
        IDENTIFICATION    string of 20 characters or 20 BCD digits used to uniquely identify the
                                device.

                          this Element is the same as UTIL_INFO_TBL.DEVICE_ID when they
                                both are available.
```

The above example depicts a relatively poorly formatted DOCUMENT, however it is valid. Clearly the effort in validating such a badly formatted document may be greater than expected, to the detriment of the CANDIDATE (e.g. this may increase the ACCREDITOR's level of effort in performing the process, and it may also increase the chance of having un-detected errors).

Therefore, it is RECOMMENDED, that the DOCUMENT SHOULD emulate the C12.19 Standard published Document Form to the highest degree practical.

3.3.2 Test DOCUMENT is Not Valid

The following document should be compared with Figure 1: Sample DOCUMENT or Standard Document Form Table Definition.

Figure 3: Sample Invalid DOCUMENT.

9.1.6 **Table 6 Device Identification Table**

Table 05 Data Description

DEVICE_IDENT_TBL (Table 05) The table provides the unique identifier for the device as specified by the user.

Global Default Table Property Overrides: Role="DATA", Atomic="true"

Table 05 Type Definitions

```
TYPE IDENT_RCD = PACKED RECORD
        IF GEN_CONFIG_TBL.ID_FORM == 0 THEN
                Identification      : BCD(10);
        ELSE
                IDENTIFICATION   : STRING(20);
        END
END;

TABLE 5 DEVICE_IDENT_TBL = IDENT_RCD;
```

Table 05 Element Descriptions

Identifier	Value	Definition
IDENT_RCD		
IDENTIFICATION		String of 20 characters or 20 BCD digits used to uniquely identify the device. This Element is the same as **UTIL_INFO_TBL.DEVICE_ID** when they are both available.

The errors are highlighted in YELLOW as follows:

1. Line 1, Found "Table 6" instead of "Table 5" in section header.
2. Line 3, Found "The" instead of "This" in Table description.
3. Line 5, Invalid property "Role", is "DATA" should be "CONTROL"
4. Line 5, Invalid extra property "Atomic"
5. Line 8, Invalid expression, expecting "!=" found "=="
6. Line 9 Invalid capitalization of "IDENTIFICATION", found "Identification"
7. Line 12, Missing ';' after "END"
8. Line 23, "are both" instead of "both are" in description of "IDENTIFICATION" member of "IDENT_RCD".

4 Requirements for a CANDIDATE's TDL Documents

4.1 General Requirements

The ANSI C12.19 Standard Document Form includes many sections. Although it is possible to create a TDL/XML Form file that may be used to render the entire C12.19 Standard as a DOCUMENT, it is not necessary to do so. It is minimally required to render the Standards' sections that contain normative syntactical definition elements and properties that must be completely and uniformly available to all AMI applications that reference, access or process the Standard TDL/XML Form file as specified in Annex I of the ANSI C12.19 Standard (Ref. [1] and Ref. [2]), and as prepared by a CANDIDATE for certification. The required TDL/XML Form content that needs to be created from the standards is identified in Table 2: Optional and mandatory sections ANSI C12.19 rendered as TDL, and in *Table 3: Optional and mandatory sections ANSI C12.22 rendered as TDL*. Optional sections are marked with an '**O**', implying that the CANDIDATE MAY include the optional sections in its TDL/XML Form DOCUMENT. Mandatory sections are marked with an '**M**', requiring that CANDIDATE must include the required sections in its TDL/XML Form. All optional and mandatory sections submitted by the CANDIDATE will be tested for conformance by the ACCREDITOR.

Table 2: Optional and mandatory sections ANSI C12.19 rendered as TDL

Standard Section #	Section Name	Contains Definitions, Syntax, Enumerations or Properties	Section is Mandatory or is Optional in TDL Document	Comment
1	Overview	No	O	
2	References	No	O	
3	Definitions	No	O	
4	General	Yes	O	Subsections contain semantically related descriptions.
4.1	Standard Tables	Yes	O	Subsections contain semantically related descriptions.
4.1.1	Standard Tables Grouping	Yes	O	All Standard Tables shall be defined inside Decade scopes, and the role attribute should be indicated for FLC and FLC+1 Tables.
4.1.2	Standard Tables Properties	Yes	O	Standard-defined default Table attributes need not be supplied explicitly in the TDL/XML file. However, when the Standard TDL/XML file provides explicit attributes then they shall match those stated by the C12.19

Standard Section #	Section Name	Contains Definitions, Syntax, Enumerations or Properties	Section is Mandatory or is Optional in TDL Document	Comment
				Standard.
4.1.3	Standard Procedure Properties	Yes	O	Standard-defined default Procedure attributes need not be supplied explicitly in the TDL/XML file. However, when the Standard TDL/XML file provides explicit attributes then they shall match those stated by the C12.19 Standard.
4.2	Manufacturer Tables	Yes	O	Not present in Standard TDLs.
4.3	Packed Record, Bit Field And Element Properties	No	O	References C12.19 Annex I where the rules are provided.
4.4	Extended User-Defined Tables Properties	Yes	O	Defines the values assigned by the Standard for the Table class attribute.
5	Syntax	No	O	References C12.19 Annex G and Standard's Document Form production rules.
6	Special Data Types	Yes	O	Subsections define the structures of "built-in" C12.19 Standard data types. These types are referenced by all Tables; therefore, they need to be defined in the TDL for proper compilation and resolution. Note: Some subsections contain TDL syntax. They need to be defined since they shall be referenced by End Device TDLs.
6.1	Character Set Selection	Yes	M	
6.2	Non-Integer Formats	Yes	M	
6.2.1	String Numbers	Yes	O	This subsection defines in plain text the encoding rules of text strings.
6.3	Date and Time Formats	Yes	M	
6.4	Common Table or Procedure Identifier Formats	No	O	Introductory text.

Standard Section #	Section Name	Contains Definitions, Syntax, Enumerations or Properties	Section is Mandatory or is Optional in TDL Document	Comment
6.4.1	TABLE_IDA_BFLD Bit Field	Yes	M	
6.4.2	TABLE_IDB_BFLD Bit Field	Yes	M	
6.4.3	TABLE_IDC_BFLD Bit Field	Yes	M	
6.4.4	SOURCE_SELECT_RCD	Yes	M	
7	Compliance & Compatibility	No	O	
8	Table Transportation Issues	No	O	Some subsections contain TDL syntax. They need to be defined since they shall be referenced by End Device TDLs.
8.2	Pending Event Description	Yes	M	
8.3	List Management Description	Yes	M	
9	Tables	Yes	M	This is where the bulk of the C12.19 Standard Tables are defined. Note: Some subsections contain TDL syntax. They need to be defined since they shall be referenced by End Device TDLs.
Annex A	Reserved Classes For Meter Equipment Manufacturers Implementing Ansi C12.19-1997 Devices	No	O	
Annex B	History & Event Log Codes	No	O	The event codes listed in Annex B.1 are enumerated in Section 9.8 Decade 7: History Log & Event Log Tables.
Annex C	Default Sets For Decade Tables	No	O	
Annex D	Indices For Partial Table Read/Write Access	No	O	
Annex E	Event Logger Implementation	No	O	
Annex F	Transformer Losses Compensation	No	O	

Standard Section #	Section Name	Contains Definitions, Syntax, Enumerations or Properties	Section is Mandatory or is Optional in TDL Document	Comment
Annex G	Document-Form Descriptive Syntax	No	O	Defines the encoding rules and visual syntax of the C12.19 Document Form. The C12.19 Standard Document Form sections that were identified with an "M" in this table were published using the rules defined in Annex G. The TDL/XML Form conversion format to TDL/Document Form is also governed by the rules in Annex G.
Annex H	Date-Time Elements Conversion Algorithm (TM_FORMAT=3 And TM_FORMAT=4)	No	O	
Annex I	XML File Format of EDL and EDL Files	No	O	Defines the encoding rules of TDL/XML Form and the mapping rules to the C12.19 Document Form.
Annex J	Universal Identifier	No	O	This information will manifest itself in the preamble of the DOCUMENT.
Annex K	Algorithms For The Conversion of Table Element Values to Engineering Units	No	O	
Annex L	Registering or Updating Device Class OID	No	O	
Annex M	Listing of Editorial Errors and Errors of Omission In ANSI C12.19-2008	No	O	Only present in IEEE 1377
Annex M	Historical Background	No	O	Only present in ANSI C12.19

Table 3: Optional and mandatory sections ANSI C12.22 rendered as TDL

Section #	Section Name	Contains Semantics, Syntax, Enumerations or Properties	Section is Mandatory or is Optional in TDL Document	Comment

Section #	Section Name	Contains Semantics, Syntax, Enumerations or Properties	Section is Mandatory or is Optional in TDL Document	Comment
Annex C	Modifications and Extensions to ANSI C12.19	No	O	ANSI C12.22 extends ANSI C12.19's Table Structure by adding new Tables and Procedures in support of network communications.
Annex C.1	Decade 12, Node Network Control Tables	Yes	M	Presented in place of ANSI C12.19 Section 9.13, "Decade 12: Reserved"
Annex C.2	Decade 13, Network Relay Control Tables	Yes	M	Presented in place of ANSI C12.19 Section 9.14 "Decade 13: Reserved"
Annex C.3	Universal ID Pattern Description of ApTitles	Yes	O	May be placed in the description of ANSI C12.19 network Section 9.13, "Decade 12, Node Network Control Tables" or "Decade 12, Node Network Control Tables" or as appropriate for documentation sake.
Annex C.4	Additions to Table 07, Procedure Initiate Table	Yes	M	Presented in ANSI C12.19 Document Form Section 9.1.8, "Table 07 Procedure Initiate Table" and Section 9.1.9 Table 08, Procedure Response Table". Also inserted in ANSI C12.19 Document Form Section 9.1.10 "Standard Procedures". Exact location of the insertion is governed by the ordinal value of the procedure number.
Annex C.5	Table 46, Extended Key Table	Yes	M	Presented in place of ANSI C12.19 Section 9.5.7, "Table 46 Reserved".
Annex C.6	Table 47, Host Access Security Table	Yes	M	Presented in place of ANSI C12.19 Section 9.5.8 "Table 47 Reserved".

4.2 TDL DOCUMENT Cover Page

The TDL top-level element contains descriptive attributes of the CANDIDATE's TDL. This information provides unique and traceable bindings between a referenced Standard document and the CANDIDATE supplied TDL file.

The ANSI C12.19 Standard does not provide explicit formatting rules for the “cover page”, so it can take on many forms. Unless otherwise specified, the minimal cover page of the TDL will include the relative Device Class: “0.2.0.0” that represents ANSI C12.19-2008 version 2.0.

The certificate issued by the ACCREDITOR should include the CANDIDATE’s TDL validation signature.

Table 4: Requirements for the production of TDL DOCUMENT Cover Page

Document Form Region	Region Description	M/O	ANSI C12.19 Reference
Cover Page	Introduces the “TDL Title Section” where the title, version, device class, standard, registry, iso branch, date and validation attributes are published.	M	Annex I.2.1 <tdl> Root Element
Title	The title of the TDL Document on the TDL title page, e.g. “ANSI C12.19-2008 Utility Industry Standard Tables”.	O	Annex I.2.1.2 <tdl> Attributes, title
Version	The version of the TDL document (referenced C12.19 Standard), e.g. “2.0”	M	Annex I.2.1.2 <tdl> Attributes, version
Device Class	The Device Class of the TDL file. Device class "0.x.y.z" is reserved for the Standard TDLs, e.g. “0.2.0.0”. All other Device Classes describe vendor specific End Devices.	M	Annex I.2.1.2 <tdl> Attributes, deviceClass
Standard	The referenced standard this TDL is based on. The referenced standard is expressed as a unique Universal Resource Indicator (URI), e.g. http://www.ansi.org/C1219TDL-2008.xml.	O	Annex I.2.1.2 <tdl> Attributes, standard
Registry	The name of the entity that is the registrar of the device class identifiers by this TDL, e.g. “REGINC”.	O	Annex I.2.1.2 <tdl> Attributes, registry
ISO Branch	The ISO branch attribute associates a registry with an ISO object directory branch, e.g. “2.16.124.113620.1.19”	O	Annex I.2.1.2 <tdl> Attributes, isoBranch
Validation	The standard requires that the registrar insert a validation text that can be used to sign the certified TDL. This signature may be used to confirm that the certified TDL is identical to the TDL that the ACCREDITOR processed and approved.	M	Annex I.2.1.2 <tdl> Attributes, validation
TDL Date	The date this TDL was last revised by the CANDIDATE.	O	Annex I.2.1.2 <tdl> Attributes, date

4.3 TDL Document Form Description Elements

The TDL Document Form description sections generically correspond to the TDL/XML Form <description> elements. They are text-formatting containers that are modeled on a restricted subset of XHTML (Ref. [10]). It provides a text formatting area to describe the parent element and inner-elements (Ref. [1] and Ref. [2], Annex I.2.2 <description> element).

There are a number of presentation formats for the description elements:

1. Presented as a description of a top-level section such as TDL-level heading, DECADE introducer, TABLE introducer, PROCEDURE introducer or built-in type introducer; or
2. Presented as an Element and Type description of data types such as PACKED RECORD, BIT FIELD, and Enumerated list; or

3. Presented as documentation within enclosing curly parenthesis (‘{‘, ‘}’) in the type definition section to describe things such as IF/THEN/ELSE constructions, SWITCH/CASE/DEFAULT constructions and definition placeholders for named Enumerators.

Forms 1 and 3 (above) are expressed in the TDL Document Form as text that can span the full width of the printed page. Form 2 manifests itself on the right-hand column of the page under the third column of the “Identifier”, “Value” and “Definition” triplet. Otherwise, there is no significant difference among the interpretation of the forms.

This section serves to identify the various common documentation formats and provide guidance to assist the ACCREDITOR recognize and distinguish plain-text documentation from Element attributes, properties, enumeration or type definitions.

4.3.1 Format 1 Documentation – Top-level Descriptions

The description text below illustrates text that is encapsulated using XHTML notation within the TDL/XML <description> element. Text that is grayed out represents generated text from the surrounding context, such as DECADE, TABLE, and PROCEDURE etc.

4.3.1.1 TDL-level General Text

> **4 General**
>
> **4.1 Standard Tables**
>
> Standard Tables are those whose structures are specified by this Standard. They 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 presently defined. This Standard provides for a total of 2040 Standard Tables, although not all are defined.

In the above text, there is no syntactical context. This text is present in the TDL top-level file context. Therefore, the headings text “**4 General**” and “**4.1 Standard Tables**” are created by the CANDIDATE using XHTML heading tag, such as <h1>General</h1> inside a description element.

4.3.1.2 TDL-level Preamble to Built-in Types

> **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; a flag indicating extended-user-defined table and two additional fields for definition and use within the defining data structure. When the **EUDT_FLAG** flag is set then **MFG_FLAG** shall be set to zero.

In the above text, there is no syntactical context. This text is present in the TDL top-level file context that leads to a definition of a build-in type. Therefore, the heading text “**6.4.1 TABLE_IDA_BFLD bit field**” is created by the CANDIDATE using XHTML heading tag, such as <h1>TABLE_IDA_BFLD bit field </h1> inside a description element.

4.3.1.3 Decade-level Descriptive Text

> **9.1 Decade 0: General Configuration Tables**
>
> Decade 0 Name
>
> GEN_CONFIG_DEC

> **Decade 0 Data Description**
>
> This decade contains tables associated with End Device configuration, identification, procedures and responses, and information required for data manipulation due to hardware configurations.

A Decade must have a name, a number and it may have additional attributes such as a label. The above example introduces a C12.19 Decade. It expresses in grayed text the Document Form members that are generated from the Decade container's attributes (name= "GEN_CONFIG_DEC", number="0" and label="General Configuration Tables").

The black text is created by the CANDIDATE using the XHTML description element to introduce the Decade.

Caveats: The name attribute of the Decade is expressed ahead of the Decade description shown here in gray. The text "…**GEN_CONFIG_DEC**…" SHOULD be generated from the Decade's name attribute, and it does not have a TDL/Document Form defining type or syntax otherwise.

4.3.1.4 Tables level Descriptive Text

> **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.
>
> **Global Default Table Property Overrides:** atomic = "true", role="CONTROL", accessibility="READONLY"

A Table must have a name, a number and it may have additional attributes such as label and access properties. The above example introduces a C12.19 Table. It expresses in grayed text the Document Form members that are generated from the Table container's attributes (name="GEN_CONFIG_TBL", number="0" and label="General Configuration Table", atomic="true", role="CONTROL" and accessibility="READONLY").

The black text is created by the CANDIDATE using the XHTML description element to document the Table.

Caveats: The name attribute of the Table is expressed only in Table definition syntax section of the DOCUMENT (not shown here).

Note: The text "**GEN_CONFIG_TBL** (Table 00)…" shown in this sample text was not generated from the Table's name attribute. i.e. is part of the description text.

4.3.1.5 Procedure level Descriptive Text

> **9.1.10.10 Procedure 09 Remote Reset**
>
> **Procedure 09 Data Description**
>
> When invoked, this procedure attempts to perform the specified combination of resets. The following resets are supported: Self-read, Demand Reset, and Season Change.

A Procedure must have a name, a number and it may have additional attributes such as label. The above example introduces a C12.19 Procedure. It expresses in grayed text the Document

Form members that are generated from the Procedure container's attributes (name="REMOTE_RESET_PROC", number= "9" and label="Remote Reset").

The black text is created by the CANDIDATE using the XHTML description element to document the Procedure.

Caveats: The name of the Procedure is used only in the defining syntax (not shown here).

4.3.2 Format 2 Documentation – Element Descriptions Sections

The description text below illustrates text that is encapsulated using XHTML notation within the TDL/XML <description> element that is part of an element syntax container, such as PACKED RECORD, BIT FIELD and Enumerator.

Generally an element's syntax description is expressed under the third column of the "Identifier", "Value" and "Definition" triplet.

e.g.

Table 05 Element Descriptions

Identifier	Value	Definition
IDENT_RCD		
	IDENTIFICATION	String of 20 characters or 20 BCD digits used to uniquely identify the device. This Element is the same as **UTIL_INFO_TBL.DEVICE_ID** when they both are available.

Text that is grayed out represents generated text from the surrounding context, such as DECADE, TABLE, PROCEDURE, PACKED RECORD, BIT FIELD, Element, Sub-element and enumerators.

The black text is created by the CANDIDATE using the XHTML description element to document the PACKED RECORD.

4.3.2.1 PACKED RECORD level Descriptive Text

The description text below illustrates text that is encapsulated using XHTML notation within the TDL/XML <description> element that is embedded in a PACKED RECORD structure.

An element's syntax description is expressed under the third column of the "Identifier", "Value" and "Definition" triplet.

e.g. 1:

TDL Element Descriptions

Identifier	Value	Definition
HTIME_DATE_RCD		High-precision (sub-second) time stamp. This is the prototype definition of the built-in type **HTIME_DATE**.

A PACKED RECORD must have a name, and it may have additional attributes and assigned properties. The above example introduces a C12.19 PACKED RECORD. It expresses in grayed text the Document Form members that are generated from the PACKED RECORD container's attributes (name="HTIME_DATE_RCD" and the parent's scope being TDL).

The black text is created by the CANDIDATE using the XHTML description element to document the PACKED RECORD.

e.g. 2:

Decade 15 Element Descriptions		
Identifier	Value	Definition
WAVEFORM_LIST_STATUS_RCD		The generic waveform capture list status management packed record. **Redefines: LP_STATUS_TBL.LP_SET_STATUS_RCD.** **None-default properties: Accessibility="READONLY". Atomic="TRUE".**

A PACKED RECORD must have a name, and it may have additional attributes and assigned properties. The above example introduces a C12.19 PACKED RECORD. It expresses in grayed text the Document Form members that are generated from the PACKED RECORD container's attributes (name="WAVEFORM_LIST_STATUS_RCD", redefine="LP_STATUS_TBL. LP_SET_STATUS_RCD" and atomic="true").

The black text is created by the CANDIDATE using the XHTML description element to document the PACKED RECORD.

Caveats: The description section of a PACKED RECORD may be followed by additional attributes such as: Replaces, Overrides, Redefines, Non-default properties, Alias and Deprecated. The Replaces, Overrides, Redefines properties may be further qualified with "reference is strict", as indicated by the standard (Ref. [1] and Ref. [2], Annex I.2.8.2 <packedRecord> Attributes).

Note: The ANSI C12.19 Standard was typeset in by a person. It seems that the required qualifier "Non-default properties:" was accidentally omitted in the re-qualification of PACKED RECORDs, BIT FIELDs and possibly elsewhere.

4.3.2.2 BIT FIELD level Descriptive Text

The description text below illustrates text that is encapsulated using XHTML notation within the TDL/XML <description> element that is embedded in a BIT FIELD structure.

An element's syntax description is expressed under the third column of the "Identifier", "Value" and "Definition" triplet.

e.g. 1:

Table 02 Element Descriptions		
Identifier	Value	Definition
W_WATER_DEVICE_BFLD		Elements that 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.

A BIT FIELD must have a name, and it may have additional attributes and assigned properties. The above example introduces a C12.19 BIT FIELD. It expresses in grayed text the Document Form members that are generated from the BIT FIELD container's attributes (name="W_WATER_DEVICE_BFLD").

The black text is created by the CANDIDATE using the XHTML description element to document the BIT FIELD.

e.g. 2:

Procedure 31 Element Descriptions

Identifier	Value	Definition
SECURED_READ_REQ_BFLD		Bit field used to indicate the secured read related operations parameters being disabled. **Redefines: START_SECURED_REGISTER_PROC. SECURED_READ_REQ_BFLD**.

The above example introduces a C12.19 BIT FIELD. It expresses in grayed text the Document Form members that are generated from the BIT FIELD container's attributes (name="SECURED_READ_REQ_BFLD" and redefine="START_SECURED_REGISTER_PROC. SECURED_READ_REQ_BFLD").

The black text is created by the CANDIDATE using the XHTML description element to document the BIT FIELD.

Caveats: The description section of a BIT FIELD may be followed by additional attributes such as: Replaces, Overrides, Redefines, Non-default properties, Alias and Deprecated. The Replaces, Overrides, Redefines properties may be further qualified with "reference is strict", as indicated by the standard (Ref. [1] and Ref. [2], Annex I.2.15.2 <bitField> Attributes).

4.3.2.3 Enumerator level Descriptive Text

The description text below illustrates text that is encapsulated using XHTML notation within the TDL/XML <description> element that is embedded in named enumerated type structure.

An element's syntax description is expressed under the third column of the "Identifier", "Value" and "Definition" triplet.

e.g. 1: named Enumerator

TDL Element Descriptions

Identifier	Value	Definition
NI_FORMAT_ENUM		Values that may be assumed by **GEN_CONFIG_TBL.FORMAT_CONTROL_3.NI_FORMAT1** (which controls the interpretation of the built-in type **NI_FMAT1**) and **GEN_CONFIG_TBL.FORMAT_CONTROL_3.NI_FORMAT2** (which controls the interpretation of the built-in type **NI_FMAT2**).
	0	**FLOAT64**

A named enumerated type must have a name that ends with an "_ENUM" suffix, and it may have additional attributes and assigned properties. The above example introduces a C12.19 Enumerator. It expresses in grayed text the Document Form members that are generated from the Enumerator's container's attributes (name="NI_FORMAT_ENUM").

The black text is created by the CANDIDATE using the XHTML description element to document the Enumerator.

e.g. 2: un-named Enumerator

Table 11 Element Descriptions

Identifier	Value	Definition
SOURCE_FLAGS_BFLD		**Redefines: DIM_SOURCES_LIM_TBL. SOURCE_FLAGS_BFLD.**
PF_EXCLUDE_FLAG		Power fail exclusion is defined as ignoring demand for maximum calculations for some period immediately after a power failure
	FALSE	Power fail exclusion is not active.
	TRUE	Power fail exclusion is active

An un-named enumerator is contained within an Element or Sub-Element (the PF_EXCLUDE_FLAG Sub-element in this example). The above example introduces a C12.19 Sub-element PF_EXCLUDE_FLAG. It expresses in grayed text the Document Form members that are generated from the Enumerator's enum or default members attributes ("FALSE" and "TRUE" in this case).

The black text is created by the CANDIDATE using the XHTML description element to document the Sub-element PF_EXCLUDE_FLAG.

Caveats: It is not clear at first glance whether the description text (in black) belongs to the Sub-element PF_EXCLUDE_FLAG or to its unnamed Enumerator. In most instances of such use by ANSI C12.19 the description is associated with the Element or Sub-element (as applicable). When the TDL quote attribute is supplied then the enumerated value must be enclosed within single or double quotes (Ref. [1] and [2], Annex I.2.5.2 <enumerator> DTD (un-named)).

4.3.3 Format 3 Documentation – Descriptions Within Type Definition Sections

The description text below illustrates text that is encapsulated using XHTML notation within the TDL/XML <description> element that is embedded inside IF/THEN/ELSE or SWITCH/CASE/DEFAULT containers. Such descriptions produce output directly in-line with the type definitions (syntax sections). i.e. these descriptions are not under the third column of the "Identifier", "Value" and "Definition" triplet.

e.g. 1: IF/THEN/ELSE descriptions

```
TYPE EXTENDED_STATUS_BFLD = BIT FIELD OF UINT8
        { This bit field is expected to be used as an entry of an array.  The interpretation
        of the nibbles of each UINT8 member of the array depends on its position index
        being the first in the collection (.lastIndex == 0) or not being the first (.lastIndex >
        0). }
        IF .lastIndex == 0 THEN
                { The index value of this bit field (i.e., the first UINT8) in the containing
                array is 0, implying that this UINT8 contains the extended common
                status flags (bits 4..7) and the extended channel status of channel 1 (bits
                0..3). }
                COMMON_DST_FLAG                 : BOOL(4);
                COMMON_POWER_FAIL_FLAG          : BOOL(5);
                COMMON_CLOCK_SET_FWD_FLAG  : BOOL(6);
                COMMON_CLOCK_SET_BKWD_FLAG: BOOL(7);
        ELSE
                { This is the description of the extended status array indices that are
                greater than zero. Each UINT8 contains data for of even channels 2,4,6
                and odd channels 3,5,7 … through channel NBR_CHNS_SETn, where n
                is the load profile set number.  The last CHANNEL_STATUS_ODD bits
                are FILL bits, when the number of channels is even. }
                CHANNEL_STATUS_EVEN             : UINT(4..7);
        END;
        CHANNEL_STATUS_ODD                      : UINT(0..3);
END;
```

The grayed text are the Document Form members that are generated from the BIT FIELD (name="EXTENDED_STATUS_BFLD" in this example) Sub-element and structure definitions.

The black text is created by the CANDIDATE using the XHTML description element to document the BIT FIELD's conditional (IF) statement, the THEN clause and the ELSE clause (Ref. [1] and [2], Annex I.2.12.3 <if> Document Form, Annex I.2.12.7 <then> Document Form, Annex I.2.13.3 <else> Document Form). All other areas that are marked in gray would be documented according to Form 1 or 2 descriptions discussed in the previous sections.

Caveats: See also Ref. [1] and [2], Annex I.2.14.3 <switch> Document Form, Annex I.2.14.7 <case> Document Form, and Annex I.2.14.11 <default> Document Form.

e.g. 2: Named Enumerator

Decade 7 Type Definitions

{ *Enumerator EVENT_CODES_ENUM* }

Decade 07 Element Descriptions

Identifier	Value	Definition
EVENT_CODES_ENUM		The following are the Standard event codes that may be entered into the history log and event log tables. For more information on their use See ANNEX B, "History & Event Log Codes" and ANNEX E, "Event Logger Implementation".
	0	No Event.

ANSI C12.19 documents its Enumerator in the Element description section in the third column of the "Identifier", "Value" and "Definition" triplet (Ref. [1] and [2], Annex I.2.6.3 <enum> Document Form).

The black text is generated by the CANDIDATE's TDL Document Form producer, thus emulating the publication behavior of the Standard.

e.g. 3: Enumerated Constants

Identifier	Value	Definition
TD_CONTROL_RCD		Control element for time-domain waveform capture. It determines the sampling characteristics and related input channels.
TIME_CAPTURE_TYPE		This type identifies the nature of the time-domain capture trace.
	0	**POINT_ON_WAVE_CNST** – digitized waveform data points.
	1	**RMS_CNST** – digitized output of RMS calculation of waveform using a one (1) cycle integration period.

ANSI C12.19 documents its Enumerated constants in the Element description section in the third column of the "Identifier", "Value" and "Definition" triplet (Ref. [1] and [2], Annex I.2.6.2 <enum> Attributes (name), Annex I.2.6.3 <enum> Document Form) and Annex G.12 Constants.

The black text is generated by the CANDIDATE's TDL Document Form producer, thus emulating the publication behavior of the Standard.

Caveats: Ref. [1] and Ref. [2], Annex G.16, "Properties", introduces the "Properties" constructor to document enumerators and similar properties in the type definitions section (syntax). However, these have not been used anywhere, despite the abundant use of Enumerators throughout the Standards (Ref [1] and [2]). In part, this may be due to the fact that the use of this syntax poses duplicity in definition. For this reason it is permissible for the CANDIDATE to produce Enumerator output in the Type Definition Section from the CANDIDATE TDL as per example 2 above.

Note: The encoding requirements for the <enumerator> in the TDL/XML Form (which is machine readable) are not affected by this rule. Therefore, it has no impact on implementation. However they are consistent with the 2011 revisions to the standard per Annex G.12, "Constants".

4.4 TDL Document Form Type Definitions

The Document Form type definitions are commonly referred to as the ANSI C12.19 Tables syntax. The TDL Document Form syntax is expressed using the pseudo Pascal (language) style semantics Ref. [6] that SHALL be generated from the CANDIDATE's TDL file. The ANSI C12.19 pseudo Pascal constructions rules are defined in Ref. [1] and Ref. [2], Annex 5, "Syntax" and Annex G, "Document-Form Descriptive Syntax". The use of type definition rules is common to all sections of the document. For this reason it is most appropriate to describe the common Document Form syntax expression in this section, the TDL Document top-level section.

The following common constructors MAY be used in any the document TDL sections (also known as document scopes or TDL contexts):

1. TDL (the document top level scope)
2. Decade (SHALL be contained within the TDL scope)
3. Table (SHALL be contained within the TDL scope or within a Decade scope)
4. Procedure (SHALL be contained within a TDL scope or within Decade scope).

   Note: Procedures SHALL be expressed in Standard Procedures section and also in Standard Table 7, “Procedure Initiate Table” section, and in the Standard Table 8, “Procedure Response Table” section, using section’s appropriate syntax.

4.4.1 Packed Record Definition

4.4.1.1 Example: Simple Packed Record Definition

The PACKED RECORD is a structure that contains Table Elements. The PACKED RECORD presentation is mapped from the TDL/XML Form <packeRecord> element. A PACKED RECORD MAY contain any the following Elements:

1. Simple Elements of any built-in type (e.g. UINT8, NI_FORMAT1) or derived type (e.g. FOO_BFLD or BAR_RCD) or sized types, such as
   a. Strings (a collection of characters, e.g. STRING(20))
   b. Binary (a collection of bytes, e.g. BINARY(30))
   c. Binary coded decimals (a collection of nibbles, e.g. BCD(8))
2. Sets (a collection of Boolean flags, e.g. SET(99)
3. Arrays (a collection of any Elements, e.g. ARRAY [20] OF UINT8)
4. Conditional (any number of IF/THEN/ELSE statements)
5. Selections (any number of CASE/SWITCH/DEFAULT statements)
6. Comments (text delimited with ‘{‘ and ‘}’

Table 5: Requirements for the production of TDL/Document Form PACKED RECORD

Document Form Region	Region Description	M/O	ANSI C12.19 Reference
Packed record syntax introducer	Header text that separate the C12.19 Packed Record syntax section from the containing scope description section, or from a description section. e.g. **Table 05 Type Definitions** **Caveats**: 1. Packed Record header text separator is expected only when the Packed Record is first type in a sequence of related type in a related collection of definitions.	O	

Packed record definition inside syntax section	Introduces a packed record syntax definition, which SHALL include the packed record's name. The packed record **name** is obtained from the <packedRecord> **name** attribute of the TDL/XML Form, and it SHALL be published between the keyword '**TYPE'**, and the keywords '= **PACKED RECORD**'. The packed record concludes with the keyword '**END**' that is followed by a semicolon '**;**'
	
	e.g.
	`TYPE FOO_RCD = PACKED RECORD`
	`… packed record elements`
	`END ;`
	
	The syntax is case sensitive and all keywords SHALL comply with the definitions found in Annex G of ANSI C12.19. These SHALL be entered exactly as written in the Standard for each type defined, using the published name.
	
	When the packed record type is defined elsewhere and redefined in the containing scope then the packed record type's wrapper and definition SHOULD be omitted from this Types Syntax section.
	
	Caveats:
	1. Font style and indentation and white space may differ from the Standard.
	2. Given that the Standard was typed in by hand and the DOCUMENT is produced by machine it is conceivable that the Standard may contain syntax error. These errors MAY be documented and explained by the CANDIDATE and SHALL be considered by the ACCREDITOR.
	3. Also see Section 4.8, "ANSI C12.19 Non-use of "Properties" in Syntax".
		M	Annex G.4 Identifiers, Annex G.11 Packed record, I.2.8 <packedRecord> Element

Document Form Region	Region Description	M/O	ANSI C12.19 Reference

Packed record element such as built-in type, derived type, STRING, BINARY or BCD definition in the syntax section	A simple packed record element SHALL be expressed by producing the element's **name**, followed by a colon ('**:**') followed by the element's **type** followed by its **length** (if the type is **STRING**, **BINARY** or **BCD**) followed by a semicolon ('**;**').
	
	e.g.
	`STD_VERSION_NO : UINT8;`
	`TOT_DATA_BLOCK : DATA_BLK_RCD;`
	`OWNER_NAME : STRING(20);`
	`DEVICE_CLASS : BINARY(4);`
	`DEVICE_ID : BCD(10);`
	
	The element's **name** and **type** SHALL be obtained from the packed record's <element> **name** and **type** attributes of the TDL/XML Form.
	
	The syntax is case sensitive and all keywords SHALL comply with the definitions found in Annex G of ANSI C12.19. These SHALL be entered exactly as written in the Standard for each type defined, using the published name.
	
	Caveat:
	1. Font style and indentation and white space may differ from the Standard.
	2. Given that the Standard was typed in by hand and the DOCUMENT is produced by machine it is conceivable that the Standard may contain syntax error. These errors MAY be documented and explained by the CANDIDATE and SHALL be considered by the ACCREDITOR.
	3. Also see Section 4.8, "ANSI C12.19 Non-use of "Properties" in Syntax".
		M	Annex G.4 Identifiers, Annex G.11 Packed record, Annex G.5.1 Basic data type definitions I.2.9 <element> Element (Child of <packedRecord>), and more specifically Annex I.2.9.3 <element> Document Form

Packed record SET element definition in the syntax section	A packed record **SET** element SHALL be expressed by producing the element's **name**, followed by a colon ('**:**') followed by its **dimension** followed by a semicolon ('**;**'). e.g. `STD_TBLS_USED : SET(NBR_TBLS);` The element's **name** and **type** SHALL be obtained from the packed record's <element> **name** and **type** attributes of the TDL/XML Form. The syntax is case sensitive and all keywords SHALL comply with the definitions found in Annex G of ANSI C12.19. These SHALL be entered exactly as written in the Standard for each type defined, using the published name. **Caveats**: 1. Font style and indentation and white space may differ from the Standard. 2. Given that the Standard was typed in by hand and the DOCUMENT is produced by machine it is conceivable that the Standard may contain syntax error. These errors MAY	M	Annex G.4 Identifiers, Annex G.11 Packed record, Annex G.5.1 Basic data type definitions I.2.9 <element> Element (Child of <packedRecord>), and more specifically Annex I.2.11.3 <set> Document Form and Annex I.2.11.2 <set> Attributes.

Document Form Region	Region Description	M/O	ANSI C12.19 Reference
	be documented and explained by the CANDIDATE and SHALL be considered by the ACCREDITOR. 3. Also see Section 4.8, "ANSI C12.19 Non-use of "Properties" in Syntax". **4. The dimension value reported in the TDL/Document Form for this type shall be 1/8 of the dimension reported in the TDL / XML Form (for more information see ANSI C12.19 Annex I.2.11.3) See also, Section 5.6.5. of this document.**		

Packed record ARRAY element definition in the syntax section	A packed record **ARRAY** element SHALL be expressed by producing the element's **name**, followed by a colon ('**:**') followed by the keyword '**ARRAY**' followed by the dimension of the array enclosed within '**[**' and '**]**', followed by the keyword '**OF**', followed by the array element's **type** and MAY follow by '(', **length,** ')' and then followed by a semicolon ('**;**').
	e.g. 1.
	`DATA_VALUES : ARRAY [5] OF UINT8;`
	e.g. 2.
	`NAMES : ARRAY [2] OF STRING(20);`
	
	The array's **name, type, dimension** and **length**, SHALL be obtained from the packed record's <array> **name, type**, **dimension** and **length** attributes of the TDL/XML Form. The **length** attribute SHALL only be expressed (exposed) when the array type is **BINARY**, **STRING, SET** or **BCD**.
	
	The syntax is case sensitive and all keywords SHALL comply with the definitions found in Annex G of ANSI C12.19. These SHALL be entered exactly as written in the Standard for each type defined, using the published name.
	
	Caveats:
	1. Font style and indentation and white space may differ from the Standard.
	2. Given that the Standard was typed in by hand and the DOCUMENT is produced by machine it is conceivable that the Standard may contain syntax error. These errors MAY be documented and explained by the CANDIDATE and SHALL be considered by the ACCREDITOR.
	3. Also see Section 4.8, "ANSI C12.19 Non-use of "Properties" in Syntax".
	4. The length reported in the TDL/Document Form for type SET shall be 1/8 of the length reported in the TDL / XML Form (see C12.19 Annex I.2.11.3).
		M	Annex G.4 Identifiers, Annex G.11 Packed record, I.2.10 <array> Element (Child of <packedRecord>), and more specifically I.2.10.3 <array> Document Form and Annex I.2.11.2 <set> Attributes.

Packed record IF, THE, ELSE use	A packed record **IF** statement SHALL be expressed by producing the keyword '**IF**' followed by the **condition** followed the keyword '**THEN**' followed by the packed record's elements, sets, or array definitions; or **IF** or **SWITCH** statements. Then it MAY be followed	M	Annex G.4 Identifiers, Annex G.11 Packed record,

Document Form Region	Region Description	M/O	ANSI C12.19 Reference

in the syntax section	by the a matching keyword '**ELSE**', followed by more packed record elements, sets, or array definitions; or **IF** or **SWITCH** statements; then it SHALL be followed by the closing (matching) keyword '**END**' followed by semicolon ('**;**').
	
	e.g. 1.
	`IF GEN_CONFIG_TBL.ID_FORM != 0 THEN`
	`IDENTIFICATION : BCD(10);`
	`ELSE`
	`IDENTIFICATION : STRING(20);`
	`END;`
	
	e.g.2.
	`IF GEN_CONFIG_TBL.TM_FORMAT != 0 THEN`
	`FW_LOAD_DATE : DATE;`
	`END;`
	
	e.g.3.
	`IF GEN_CONFIG_TBL.STD_VERSION_NO >= 2 THEN`
	`  IF GEN_CONFIG_TBL.TM_FORMAT != 0 THEN`
	`  FW_LOAD_DATE : DATE;`
	`  END;`
	`END;`
	
	The **IF** statement's **condition** SHALL be obtained from the packed record's <if> **condition** attribute of the TDL/XML Form.
	
	The syntax is case sensitive and all keywords SHALL comply with the definitions found in Annex G of ANSI C12.19. These SHALL be entered exactly as written in the Standard for each type defined, using the published name.
	
	Caveats:
	Font style and indentation and white space may differ from the Standard.
	The **condition** MAY be surrounded by parenthesis ('(' and ')').
	The keywords '**FALSE**' and '**TRUE**' within the **condition** of the **IF** statement MAY be substituted with the keywords '**false**' and '**true**' respectively.
	Given that the Standard was typed in by hand and the DOCUMENT is produced by machine it is conceivable that the Standard may contain syntax error. These errors MAY be documented and explained by the CANDIDATE and SHALL be considered by the ACCREDITOR.
	Also see Section 4.8, "ANSI C12.19 Non-use of "Properties" in Syntax".		

Packed record SWITCH, CASE, DEFAULT	A packed record **SWITCH** statement SHALL be expressed by producing the keyword '**SWITCH**' followed by the **selection** followed the keyword '**OF**' followed by the zero or more **CASE** statements; that MAY be followed by a single **DEFAULT** statement; then it SHALL be followed the closing (matching) keyword '**END**' and	M	Annex G.4 Identifiers, Annex G.11 Packed record, I.2.14

Document Form Region	Region Description	M/O	ANSI C12.19 Reference

use in the syntax section	followed by semicolon (‘**;**’).
	
	A **CASE** statement SHALL be expressed by producing the keyword ‘**CASE**’ followed by a range that is defined by a **startValueInclusive** value, which MAY be followed by ‘**..**’ and an **endValueInclusive** value; then it SHALL be followed by a colon ‘**:**’ followed by the packed record’s elements, sets, or array definitions; or **IF** or **SWITCH** statements.
	
	A **DEFAULT** statement SHALL be expressed by producing the keyword ‘**DEFAULT**’ followed by a colon ‘**:**’ followed by the packed record’s elements, sets, or array definitions; or **IF** or **SWITCH** statements.
	
	e.g.
	`SWITCH GEN_CONFIG_TBL.NAMEPLATE_TYPE OF`
	`CASE 0 :`
	`G_GAS_DEVICE :`
	`G_GAS_DEVICE_RCD;`
	`CASE 1 :`
	`W_WATER_DEVICE :`
	`W_WATER_DEVICE_BFLD;`
	`CASE 2 :`
	`E_ELECTRIC_DEVICE :`
	`E_ELECTRIC_DEVICE_RCD;`
	`CASE 3..255 :`
	`G_DEVICE_DESC : UINT8;`
	`G_GENERIC_DEVICE :`
	`G_GENERIC_DEVICE_RCD;`
	
	`DEFAULT:`
	`RESERVED : NIL;`
	`END;`
	
	The statement’s **selection** SHALL be obtained from the packed record’s <switch> **selection** attribute of the TDL/XML Form.
	
	The statement’s **startValueInclusive** and **endValueInclusive** SHALL be obtained from each of the packed record’s <case> **startValueInclusive** and **endValueInclusive** attributes that are contained in the <switch> element according to the TDL/XML Form.
	
	The syntax is case sensitive and all keywords SHALL comply with the definitions found in Annex G of ANSI C12.19. These SHALL be entered exactly as written in the Standard for each type defined, using the published name.
	
	Caveat:
	1. Font style and indentation and white space may differ from the Standard.
	2. The **selection**, **startValueInclusive** and **endValueInclusive** MAY be surrounded by parenthesis (‘(‘ and ‘)’).
	3. The keywords ‘**FALSE**’ and ‘**TRUE**’ within **selection**, **startValueInclusive** and **endValueInclusive** of the **SWITCH**
			<switch>Element (Child of <packedRecord>), and more specifically I.2.14.3 <switch>Document Form.

Document Form Region	Region Description	M/O	ANSI C12.19 Reference
	statement may be substituted with the keywords '**false**' and '**true**' respectively. 4. Given that the Standard was typed in by hand and the DOCUMENT is produced by machine it is conceivable that the Standard may contain syntax error. These errors MAY be documented and explained by the CANDIDATE and SHALL be considered by the ACCREDITOR. 5. Also see Section 4.8, "ANSI C12.19 Non-use of "Properties" in Syntax".		

```
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;
        DEVICE_CLASS              : BINARY(4);
        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;
```

4.4.1.2 Example: Packed Record Definition with IF/THEN/ELSE Constraints

```
TYPE MANUFACTURER_IDENT_RCD = PACKED RECORD
        MANUFACTURER                    : STRING(4);
        ED_MODEL                        : STRING(8);
        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       : BCD(8);
        ELSE
                MFG_SERIAL_NUMBER       : STRING(16);
        END;
END;
```

4.4.1.3 Example: Packed Record Definition with SWITCH/CASE/DEFAULT Constraints

```
TYPE DEVICE_DEFINITION_RCD = PACKED RECORD
        SWITCH GEN_CONFIG_TBL.NAMEPLATE_TYPE OF
                CASE 0 :
                        G_GAS_DEVICE         : G_GAS_DEVICE_RCD;
                CASE 1 :
                        W_WATER_DEVICE     : W_WATER_DEVICE_BFLD;
                CASE 2 :
                        E_ELECTRIC_DEVICE :  E_ELECTRIC_DEVICE_RCD;
                CASE 3..255:
                        G_GENERIC_DEVICE  :  G_GENERIC_DEVICE_RCD;
                DEFAULT :
                        RESERVED              : NIL;
        END;
END;
```

4.4.2 Bit Field Definition

The BIT FIELD is a structure that contains compacted Table Elements. The BIT FIELD presentation is mapped from the TDL/XML Form <bitField> element. A BIT FIELD MAY contain any the following Sub-elements:

1. Unsigned integers (i.e. UINT)
2. Signed integers (i.e. INT)
3. Booleans (i.e. BOOL)
4. Fillers (i.e. FILL)
5. Conditional (any number of IF/THEN/ELSE statements)
6. Selections (any number of CASE/SWITCH/DEFAULT statements)
7. Comments (text delimited with '{' and '}'

Table 6: Requirements for the production of TDL/Document Form BIT FIELD

Document Form Region	Region Description	M/O	ANSI C12.19 Reference
Bit Field syntax introducer	Header text that separate the C12.19 Bit Field syntax section from the containing scope description section, or from a description section. e.g. **TDL Type Definitions** **Caveats**: 1. Bit field header text separator is expected only when the Bit field is first type in a sequence of related type in a related collection of definitions.	O	
Bit Field definition inside syntax section	Introduces a bit field syntax definition, which SHALL include the bit field's **name** and **type**. The bit field **name** is obtained from the <bitField> **name** and **type** attributes of the TDL/XML Form. The **name** SHALL be published between the keyword '**TYPE'**, and the	M	Annex G.4 Identifiers, Annex G.8 Bit field, I.2.15 <BitField> Element

Document Form Region	Region Description	M/O	ANSI C12.19 Reference

	keywords '**BIT FIELD OF**' and the type published after '**BIT FIELD OF**'. The bit field concludes with the keyword '**END**' that is followed by a semicolon '**;**'
	
	e.g.
	`TYPE FOO_BFLD = BIT FIELD OF UINT8`
	`… bit field sub-elements`
	`END ;`
	
	The syntax is case sensitive and all keywords SHALL comply with the definitions found in Annex G of ANSI C12.19. These SHALL be entered exactly as written in the Standard for each type defined, using the published name.
	
	When the bit field type is defined elsewhere and redefined in the containing scope then the bit field type's wrapper and definition SHOULD be omitted from this Types Syntax section.
	
	Caveats:
	1. Font style and indentation and white space may differ from the Standard.
	2. Given that the Standard was typed in by hand and the DOCUMENT is produced by machine it is conceivable that the Standard may contain syntax error. These errors MAY be documented and explained by the CANDIDATE and SHALL be considered by the ACCREDITOR.
	3. Also see Section 4.8, "ANSI C12.19 Non-use of "Properties" in Syntax".		

Bit field sub-element definition in the syntax section	A bit field sub-element SHALL be expressed by producing the sub-element's **name**, followed by a colon ('**:**') followed by the sub-element's **type** followed by its bit range (**startBitInclusive** and **endBitInclusive**) followed by a semicolon ('**;**'). The range will be surrounded by parenthesis '(' and ')' and delimited by '**..**'. The '**..**' and the **endBitInclusive** MAY be omitted if the range is a single bit.
	e.g.
	`DEMAND_RESET_FLAG : BOOL(0);`
	`SELF_READ_FLAG : BOOL(1);`
	`SEASON_CHANGE_FLAG : BOOL(2);`
	`NEW_SEASON : UINT(3..6);`
	`FILLER : FILL(7..7);`
	
	The subElement's **name**, **type**, **startBitInclusive**, and **endBitInclusive**. SHALL be obtained from the bit field
		M	Annex G.4 Identifiers, Annex G.8 Bit field, Annex G.5.1 Basic data type definitions I.2.16 <subElement> Element (Child of <bitField>), and more specifically Annex I.2.16.3 <subElement> Document Form

Document Form Region	Region Description	M/O	ANSI C12.19 Reference
	<subElement> **name**, **type**, **startBitInclusive**, and **endBitInclusive** attributes of the TDL/XML Form. The syntax is case sensitive and all keywords SHALL comply with the definitions found in Annex G of ANSI C12.19. These SHALL be entered exactly as written in the Standard for each type defined, using the published name. **Caveats**: 1. Font style and indentation and white space may differ from the Standard. 2. Given that the Standard was typed in by hand and the DOCUMENT is produced by machine it is conceivable that the Standard may contain syntax error. These errors MAY be documented and explained by the CANDIDATE and SHALL be considered by the ACCREDITOR. 3. See Section 4.8, "ANSI C12.19 Non-use of "Properties" in Syntax".		
Bit field IF, THE, ELSE use in the syntax section	A bit field **IF** statement SHALL be expressed using the same rules for expressing packed record **IF** statements (see Section 4.4.1, "Packed Record Definition").	M	Annex G.4 Identifiers, Annex G.8 Bit field, I.2.17 <if> Element (Child of <bitField>), and more specifically I.2.17.3 <if> Document Form.
Packed record SWITCH, CASE, DEFAULT use in the syntax section	A bit field **SWITCH** statement SHALL be expressed using the same rules for expressing packed record **SWITCH** statements (see Section 4.4.1, "Packed Record Definition").	M	Annex G.4 Identifiers, Annex G.8 Bit field, I.2.20 <switch>Element (Child of <bitField>), and more specifically I.2.20.3 <switch>Document Form.

4.4.2.1 Example: Simple Bit Field Definition

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

4.4.2.2 Example: Bit Field Definition with IF/THEN/ELSE Constraints

```
TYPE SOURCE_FLAGS_BFLD = BIT FIELD OF UINT8
        PF_EXCLUDE_FLAG          : BOOL(0);
        RESET_EXCLUDE_FLAG       : BOOL(1);
        BLOCK_DEMAND_FLAG        : BOOL(2);
        SLIDING_DEMAND_FLAG      : BOOL(3);
        THERMAL_DEMAND_FLAG      : BOOL(4);
        SET1_PRESENT_FLAG        : BOOL(5);
        SET2_PRESENT_FLAG        : BOOL(6);
        IF GEN_CONFIG_TBL.STD_VERSION_NO < 2 THEN
                FILLER                   : FILL(7..7);
        ELSE
                CONVERSION_ALG_FLAG      : BOOL(7);
        END;
END;
```

4.4.2.3 Example: Bit Field Definition with SWITCH/CASE/DEFAULT Constraints

```
TYPE RDATE_BFLD = BIT FIELD OF UINT16
        MONTH         : UINT(0..3);
        SWITCH .MONTH OF
                CASE 0 :
                        PERIOD_IN_MINUTES        : UNIT(4..15);
                CASE 1..13 :
                        OFFSET                   : UINT(4..7);
                        WEEKDAY                  : UINT(8..10);
                        DAY                      : UINT(11..15);
                CASE 14 :
                        FILLER1                  : FILL(4..7);
                        WEEKDAY                  : UINT(8..10);
                        FILLER2                  : FILL(11..15);
                CASE 15 :
                        PERIOD_IN_DAYS           : UINT(4..9);
                        DELTA                    : UINT(10..15);
        END;
END;
```

Note: In the above example the line that contains "**SWITCH** .MONTH **OF**" has a space character between the SWITCH and the dot ('.'). This space is REQUIRED according to the construction rules of a **SWITCH** statement. i.e. the correct interpretation of this line is: **SWITCH** on ***dot*-MONTH OF**. An incorrect interpretation is **SWITCH.MONTH OF**.

4.5 Required Elements in a Document Form of Decade Definitions

Each expression of the DOCUMENT's Decade SHOULD begin with the Decade's descriptive text stipulated in Section 4.3.1.3, "Decade-level Descriptive Text". The Decade **name**, **number** and **label** SHALL be obtained from the TDL/XML Form <decade> **name**, **number** and **label** attributes and SHALL precede the Decade's description and type definitions. When present, the Decade **description** SHALL be obtained from the TDL/XML Form <decade>/<description> element and it SHALL follow the Decade's **name**, **number** and **label** in the DOCUMENT.

The Decade-level type definition syntax SHALL follow and SHALL be expressed in accordance with the requirements stipulated in Section 4.4, "TDL Document Form Type Definitions".

The Decade-level type descriptions SHALL follow the Decade type definitions and SHALL be expressed in accordance with the requirements stipulated in Section 4.3, "TDL Document Form Description Elements", as applicable to Decades.

The Decade SHALL expose all of the contained Tables in accordance with the Table definition that are published in the referenced Standard (e.g. Ref. [1], Ref. [2], Ref. [3] and Ref. [4]).

4.6 Required Elements in a Document Form of Table Definitions

Each expression of the DOCUMENT's Table SHOULD begin with the Table's descriptive text stipulated in Section 4.3.1.4, "Tables level Descriptive Text". The Table **name**, **number** and **label** SHALL be obtained from the TDL/XML Form <table> **name**, **number** and **label** attributes and SHALL precede the Table description and type definitions. When present, the Table **description** SHALL be obtained from the TDL/XML Form <table>/<description> element and it SHALL follow the Table's **name**, **number** and **label** in the DOCUMENT.

The Table-level type definition syntax SHALL follow and SHALL be expressed in accordance with the requirements stipulated in Section 4.4, "TDL Document Form Type Definitions".

The Table-level type descriptions SHALL follow the Table type definitions and SHALL be expressed in accordance with the requirements stipulated in Section 4.3, "TDL Document Form Description Elements", as applicable to Tables.

The Tables SHALL expose all of the contained non-default properties, qualifications and auto-generated syntax for Standard Table 7 and Standard Table 8, in accordance with the Table definitions that are published in the referenced Standard (e.g. Ref. [1], Ref. [2], Ref. [3] and Ref. [4]).

For more requirements see Section 3.2, Required Elements in a Document Form of a Table Definition.

4.7 Required Elements in a Document Form of Procedure Definitions

Procedures represent actions that can be performed by the End Device. A procedure is initiated by writing to table 7 and the result is retrieved by reading table 8. The packed record that is pointed to by the REQUEST member of the PROCEDURE statement defines the content of the PARM_RCD (procedure parameters record). This packed record SHALL also be referenced in the syntax section of Table 7 of the DOCUMENT and presented ordered sequentially in ascending order by procedure number. Similarly, the packed record that is pointed to the RESPONSE member of the PROCEDURE statement defines the RESP_DATA_RCD (response data record). This packed record SHALL also be referenced in the syntax section of Table 8 of the DOCUMENT and presented ordered sequentially in ascending order by procedure number. These are each expressed separately in the DOCUMENT section that defines the procedures, each in the local context of the corresponding Procedure. Therefore, when inspecting the TDL/XML Form definitions within Table 7 and Table 8, the ACCREDITOR SHALL NOT find any definitions of Table 7's PARM_RCD and Table 8's RESP_DATA_RCD.

> **Note**: This behavior of the TDL/XML Form to DOCUMENT conversion is referred to as auto-generate syntax for Standard Table 7 (Procedure Initiate) and Standard Table 8 (Procedure Response).

Each expression of the DOCUMENT's Procedure SHOULD begin with the Procedure's descriptive text stipulated in Section 4.3.1.5, "Procedure level Descriptive Text". The Procedure **name**, **number** and **label** attributes SHALL be obtained from the TDL/XML Form <procedure> element's attributes **name**, **number** and **label** and SHALL precede the Procedure description and type definitions. When, the Procedure **description** is present it SHALL be obtained from the

TDL/XML Form <procedure>/<description> element and it SHALL follow the Procedure's **name**, **number** and **label** in the DOCUMENT.

The Procedure's type definition syntax SHALL follow and SHALL be expressed in accordance with the requirements stipulated in Section 4.4, "TDL Document Form Type Definitions".

The Procedure's type descriptions SHALL follow the Procedure's type definitions and SHALL be expressed in accordance with the requirements stipulated in Section 4.3, "TDL Document Form Description Elements", as applicable to each Procedure.

The Procedure descriptions of the DOCUMENT SHALL expose all of the contained non-default properties and qualifications. The collation of all procedures section declarations under a single common section heading (other than Table 7 and Table 8 sections) is OPTIONAL. It is RECOMMENDED that all procedures declarations SHOULD be collated under a single heading in the DOCUMENT and placed after the section that lists Table 8. However, the DOCUMENT SHALL include auto-generated syntax for all procedure requests in the definition of Standard Table 7 (Procedure Initiate) and all procedure responses in the definition of Standard Table 8 (Procedure Response), as described above. These SHALL be derived from the TDL/XML Form <procedure> elements in accordance with the Table definitions that are published by the ANSI C12.19 Standards, Ref. [1], Ref. [2], Ref. [3] and Ref. [4].

Note: Sections 4.7.1, 4.7.2 and 4.7.3 below provide examples for the production of Procedures in the DOCUMENT from the CANDIDATE's submitted TDL/XML Form. They are provided to illustrate the fact that the CANDIDATE's TDL/XML Form Tables 7 and 8 SHALL not include type definitions inside STD_PARM_RCD or STD_RESP_RCD. The content (other than top-level descriptions) of the STD_PARM_RCD or STD_RESP_RCD SHALL be declared (for each procedure separately) only in the TDL/XML <procedure> elements (See Annex G.14 "Procedures", of Ref. [1] and Ref. [2]).

4.7.1 Example: Procedure 0 Definition without parameters

4.7.1.1 Example DOCUMENT Content in Section Standard Procedures

9.1.10.1 Procedure 00 Cold Start

Procedure 00 Data Description

This procedure causes the End Device to return to a manufacturer default known state. All data values, programs and conditions may be lost. Communication may be broken. If an Event Log exists, an attempt shall be made to retain it.

Procedure 00 Type Definitions

PROCEDURE 0 COLD_START_PROC;

…

4.7.1.2 Example DOCUMENT Content in Section Table 7, Procedure Initiate

```
TYPE STD_PARM_RCD = PACKED RECORD
  …
        IF PROC.TABLE_PROC_NBR == COLD_START_PROC THEN
              PROC_DATA_0 : NIL;
        END;
  …
END;
```

4.7.1.3 Example DOCUMENT Content in Section Table 8, Procedure Response

```
TYPE STD_RESP_RCD = PACKED RECORD
  …
        IF PROC.TABLE_PROC_NBR == COLD_START_PROC THEN
              PROC_DATA_0 : NIL;
        END;
  …
END;
```

4.7.1.4 Possible TDL/XML Form Source Definition for Procedure 0 with No Parameters

```
<procedure name="COLD_START_PROC" number="0" label="Cold Start">
        <description>This procedure causes the End Device to return to a
         manufacturer default known state.  All data values, programs and conditions
        may be lost.  Communication may be broken.  If an Event Log exists, an attempt
        shall be made to retain it.</description>
</procedure>
```

4.7.2 Example: Procedure 4 Definition with Request parameters

4.7.2.1 Example DOCUMENT Content in Section Standard Procedures

9.1.10.5 Procedure 04 Reset List Pointers

Procedure 04 Data Description

When this procedure is invoked, the End Device attempts to reset list control Elements to their initial state. To execute this procedure, the initiator shall in addition be required to have access permission to the procedure and write access permission to the table containing the selected list(s).

Procedure 04 Type Definitions

```
TYPE PARM_DATA_RCD  = PACKED RECORD
        LIST    : UINT8;
END;

PROCEDURE 4 RESET_LIST_POINTERS_PROC
        REQUEST = PARM_DATA_RCD;

…
```

4.7.2.2 Example DOCUMENT Content in Section Table 7, Procedure Initiate

```
TYPE STD_PARM_RCD = PACKED RECORD
  …
        IF PROC.TABLE_PROC_NBR == RESET_LIST_POINTERS_PROC THEN
              PROC_DATA_4 :
                    RESET_LIST_POINTERS_PROC.PARM_DATA_RCD;
        END;
  …
END;
```

4.7.2.3 Example DOCUMENT Content in Section Table 8, Procedure Response

```
TYPE STD_RESP_RCD = PACKED RECORD
  …
        IF PROC.TABLE_PROC_NBR == RESET_LIST_POINTERS_PROC THEN
              PROC_DATA_4 : NIL;
        END;
  …
END;
```

4.7.2.4 Possible TDL/XML Form Source Definition for Procedure 4 with No Parameters

```
<procedure name="RESET_LIST_POINTERS_PROC"
        number="4" label="Reset List Pointers">
        <description>When this procedure is invoked, the End Device attempts to reset
        list control Elements to their initial state.  To execute this procedure, the initiator
        shall in addition be required to have access permission to the procedure and
        write access permission to the table containing the selected list(s).
        </description>
        <packedRecord name="PARM_DATA_RCD">
              <element name="LIST" type="UINT8" />
        </packedRecord>
        <extend
              type="PARM_DATA_RCD"
              target="PROC_INITIATE_TBL.STD_PARM_RCD" />
</procedure>
```

4.7.3 Example: Procedure 6 Definition with Response parameters

4.7.3.1 Example DOCUMENT Content in Section Standard Procedures

9.1.10.7 Procedure 06 Change End Device Mode

Procedure 06 Data Description

This procedure changes the operational modes of the End Device.

Procedure 06 Type Definitions

```
TYPE PARM_DATA_RCD = PACKED RECORD
        ED_MODE                 : ED_MODE_STATUS_TBL.ED_MODE_BFLD;
END;

TYPE RESP_DATA_RCD = PACKED RECORD
        ED_MODE                 : ED_MODE_STATUS_TBL.ED_MODE_BFLD;
END;

PROCEDURE 6 CHANGE_END_DEVICE_MODE_PROC
        REQUEST = PARM_DATA_RCD
        RESPONSE = RESP_DATA_RCD;
…
```

4.7.3.2 Example DOCUMENT Content in Section Table 7, Procedure Initiate

```
TYPE STD_PARM_RCD = PACKED RECORD
  …
        IF PROC.TABLE_PROC_NBR == CHANGE_END_DEVICE_MODE_PROC
        THEN
                PROC_DATA_6 :
                        CHANGE_END_DEVICE_MODE_PROC.PARM_DATA_RCD;
        END;
  …
END;
```

4.7.3.3 Example DOCUMENT Content in Section Table 8, Procedure Response

```
TYPE STD_RESP_RCD = PACKED RECORD
  …
        IF PROC.TABLE_PROC_NBR == CHANGE_END_DEVICE_MODE_PROC
        THEN
                PROC_DATA_6 :
                        CHANGE_END_DEVICE_MODE_PROC.RESP_DATA_RCD;
        END;
  …
END;
```

4.7.3.4 Possible TDL/XML Form Source Definition for Procedure 6 with No Parameters

```
<procedure
        name="CHANGE_END_DEVICE_MODE_PROC"
        number="6" label="Change End Device Mode">
        <description>This procedure changes the operational modes of the End
        Device.</description>
        <packedRecord name="PARM_DATA_RCD">
                <element
                        name="ED_MODE"
                         type="ED_MODE_STATUS_TBL.ED_MODE_BFLD">
                        <description>…</description>
                </element>
        </packedRecord>
        <packedRecord name="RESP_DATA_RCD">
                <element
                        name="ED_MODE"
                        type="ED_MODE_STATUS_TBL.ED_MODE_BFLD">
                        <description>...</description>
                </element>
        </packedRecord>
        <extend
                type="PARM_DATA_RCD"
                target="PROC_INITIATE_TBL.STD_PARM_RCD" />
        <extend
                type="RESP_DATA_RCD"
                target="PROC_RESPONSE_TBL.STD_RESP_RCD" />
</procedure>
```

4.8 ANSI C12.19 Non-use of "Properties" in Syntax Sections

ANSI C12.19 (Ref. [1] and Ref. [2]), Annex G.16, "Properties", provides a prescription for the transcription a subset of the TDL/XML Form properties into the syntax sections of a DOCUMENT. The capabilities provide for the transcription of descriptions, labels, assertions and enumerator. The properties may be applied to Tables, Procedures, Packed records, and Bit fields.

There are two uses for the properties:

1. Used in-line at the time of definition of a type, data element or enumerator.
2. Used to qualify a predefined a Table, Procedure or a type

The ANSI C12.19 Standard does not make any use the Annex G.16 "Properties" syntax in the published Document Form. Therefore the CANDIDATE MAY include content in a DOCUMENT in accordance with Annex G.16, but this SHOULD NOT be required.

The use of Annex G.16, Properties" in the syntax section SHALL NOT be a substitute for the Standard's Document Form description sections.

The ANSI C12.19 Standard has limited vocabulary in Annex G.16 for expressing TDL/XML Form element attributes and qualifications in terms of the Document Form "Properties" in syntax section. For this reason it is RECOMMENDED that the use of the syntax defined in Annex G.16 SHOULD be limited only to the specific manner that is used by the ANSI C12.19 Standard.

5 *Testing* Exercises for ACCREDITORs

This section provides a number of practice exercises that may be used to train the ACCREDITOR's tester. The lines are numbered for ease of reference. Each exercise is divided into five sections:

Exercise Sections

1. Extracted Section from the Referenced Standard Document Form
2. Extracted Section from the CANDIDATE DOCUMENT to Compare
3. Section from the CANDIDATE TDL/XML Form used to Produce the CANDIDATE DOCUMENT

Solution Sections

4. Solution – Section from the CANDIDATE TDL/XML Form used to Produce the CANDIDATE DOCUMENT
5. Solution – Issues to be identified and reported by the ACCREDITOR

The first section presents text that was extracted from the referenced standard. The second section contains the corresponding text that was produced by the ACCREDITOR from the CANDIDATE's TDL/XML Form file, using the CANDIDATE's tool. Sections 1 and 2 should be compared by the tested, as part of the exercise. The tester should compile a list defects based on the comparison results. After the defect list is completed, the tester may wish to consult the thirst section, and see if it is consistently represent the controls as expressed in the first and second section of the exercise. The most likely discovery will be that a control element is expressed as documentation instead of control. This may apply to table names, numbers, labels, default attributes. This may also apply to packed records, bit fields, enumerators and elements attributes, names, aliases overrides etc.

The fourth and fifth section presents the list of defects and issues that should be identified by the tester. These are highlighted in yellow and detailed on a line by line basis.

The trainer should print each of these sections and present only the first three sections to the trainee. Upon completion of the exercise the solutions should be presented to the trainee, compared and discussed.

5.1 Exercise 1.

5.1.1 Extracted Section from the Referenced Standard Document Form

9.3.8 Table 27 Present Register Selection Table

Table 27 Data Description

PRESENT_REGISTER_SELECT_TBL (Table 27) provides selections to sources of present demands and present values. These indices point towards array elements in **SOURCES_TBL** (Table 16) when Table 16 is present. The values selected are available in **PRESENT_REGISTER_DATA_TBL** (Table 28). Table 27 is used to select sources for Table 28, which is continually updated.

Global Default Table Property Overrides: Role="CONTROL"

Table 27 Type Definitions

```
TYPE PRESENT_REGISTER_SELECT_RCD = PACKED RECORD
        PRESENT_DEMAND_SELECT : ARRAY[ACT_REGS_TBL.NBR_PRESENT_DEMANDS] OF
                STD.SOURCE_SELECT_RCD;
        PRESENT_VALUE_SELECT :
                ARRAY[ACT_REGS_TBL.NBR_PRESENT_VALUES] OF
                STD.SOURCE_SELECT_RCD;
END;

TABLE 27 PRESENT_REGISTER_SELECT_TBL = PRESENT_REGISTER_SELECT_RCD;
```

Table 27 Element Descriptions

Identifier	Value	Definition
PRESENT_REGISTER_SELECT_RCD		
PRESENT_DEMAND_SELECT		Array that contains data source selectors for quantities used to measure present demand. The present demand values shall be reported at the same index position in **PRESENT_REGISTER_DATA_TBL. PRESENT_DEMAND** (Table 28).
PRESENT_VALUE_SELECT		Array that contains data source selectors for quantities used to measure present value. The present values shall be reported at the same index position in **PRESENT_REGISTER_DATA_TBL.PRESENT_VALUES** (Table 28).

5.1.2 Extracted Section from the CANDIDATE DOCUMENT to Compare

9.3.8 Table 28 Present Register Selection Table

Table 28 Data Description

PRESENT_REGISTER_SELECT_TBL (Table 27) provides selections to sources of present demands and present values. These indices point towards array elements in **SOURCES_TBL** (Table 16) when Table 16 is present. The values selected are available in **PRESENT_REGISTER_DATA_TBL** (Table 28). Table 27 is used to select sources for Table 28, which is continually updated.

Global Default Table Property Overrides: Role="CONTROL"

Table 28 Type Definitions

```
TYPE PRESENT_REGISTER_SELECT_RCD = PACKED RECORD
        PRESENT_DEMANDS_SELECT : ARRAY[ACT_REGS_TBL.NBR_PRESENT_DEMANDS] OF
                SOURCE_SELECT_RCD;
        PRESENT_VALUE_SELECT :
                ARRAY[ACT_REGS_TBL.NBR_PRESENT_VALUES] OF
                SOURCE_SELECT_RCD;
END;

TABLE 28 PRESENT_REGISTER_SELECT_TBL = PRESENT_REGISTER_SELECT_RCD;
```

Table 27 Element Descriptions

Identifier	Value	Definition
PRESENT_REGISTER_SELECT_RCD		
PRESENT_DEMAND_SELECT		Array that contains data source selectors for quantities used to measure present demand. The present demand values shall be reported at the same index position in **PRESENT_REGISTER_DATA_TBL. PRESENT_DEMAND** (Table 28).
PRESENT_VALUE_SELECT		Array that contains data source selectors for quantities used to measure present value. The present values shall be reported at the same index position in **PRESENT_REGISTER_DATA_TBL.PRESENT_VALUES** (Table 28).

5.1.3 Section from the CANDIDATE TDL/XML Form used to Produce the CANDIDATE DOCUMENT

```
[1]  …
[2]
[3]  <table name="PRESENT_REGISTER_SELECT_TBL"  number="28"
[4]       type="PRESENT_REGISTER_SELECT_RCD" label="Present Register Selection Table" role="CONTROL">
[5]       <description><b>PRESENT_REGISTER_SELECT_TBL</b> (Table 27) provides selections to sources of
[6]            present demands and present values. These indices point towards array elements in
[7]            <b>SOURCES_TBL</b> (Table 16) when Table 16 is present. The values selected are available in
[8]            <b>PRESENT_REGISTER_DATA_TBL</b> (Table 28). Table 27 is used to select sources for Table
[9]            28, which is continually updated.
[10]      </description>
[11]      <packedRecord name="PRESENT_REGISTER_SELECT_RCD">
[12]        <array name="PRESENT_DEMANDS_SELECT" type=" SOURCE_SELECT_RCD"
[13]          dimension="ACT_REGS_TBL.NBR_PRESENT_DEMANDS">
[14]          <description>Array that contains data source selectors for quantities used to measure present demand.
[15]            The present demand values shall be reported at the same index position in
[16]            <b>PRESENT_REGISTER_DATA_TBL. PRESENT_DEMAND</b> (Table 28).
[17]          </description>
[18]        </array>
[19]        <array name="PRESENT_VALUE_SELECT" type="SOURCE_SELECT_RCD"
[20]          dimension="ACT_REGS_TBL.NBR_PRESENT_VALUES">
[21]          <description>Array that contains data source selectors for quantities used to measure present value. The
[22]            present values shall be reported at the same index position in <b>PRESENT_REGISTER_DATA_TBL.
[23]            PRESENT_VALUES</b> (Table 28).
[24]          </description>
[25]        </array>
[26]      </packedRecord>
[27] </table>
[28]
[29] …
[30]
```

5.1.4 Solution – Issues to be identified and reported by the ACCREDITOR

9.3.8 Table 28 Present Register Selection Table

Table 28 Data Description

PRESENT_REGISTER_SELECT_TBL (Table 27) provides selections to sources of present demands and present values. These indices point towards array elements in **SOURCES_TBL** (Table 16) when Table 16 is present. The values selected are available in **PRESENT_REGISTER_DATA_TBL** (Table 28). Table 27 is used to select sources for Table 28, which is continually updated.

Global Default Table Property Overrides: Role="CONTROL"

Table 28 Type Definitions

```
TYPE PRESENT_REGISTER_SELECT_RCD = PACKED RECORD
        PRESENT_DEMANDS_SELECT : ARRAY[ACT_REGS_TBL.NBR_PRESENT_DEMANDS] OF
                SOURCE_SELECT_RCD;
        PRESENT_VALUE_SELECT :
                ARRAY[ACT_REGS_TBL.NBR_PRESENT_VALUES] OF
                SOURCE_SELECT_RCD;
END;

TABLE 28 PRESENT_REGISTER_SELECT_TBL = PRESENT_REGISTER_SELECT_RCD;
```

Table 27 Element Descriptions

Identifier	Value	Definition
PRESENT_REGISTER_SELECT_RCD		
PRESENT_DEMAND_SELECT		Array that contains data source selectors for quantities used to measure present demand. The present demand values shall be reported at the same index position in **PRESENT_REGISTER_DATA_TBL. PRESENT_DEMAND** (Table 28).
PRESENT_VALUE_SELECT		Array that contains data source selectors for quantities used to measure present value. The present values shall be reported at the same index position in **PRESENT_REGISTER_DATA_TBL.PRESENT_VALUES** (Table 28).

5.1.5 Solution – Section from the CANDIDATE TDL/XML Form used to Produce the CANDIDATE DOCUMENT

```
[1]  …
[2]
[3]  <table name="PRESENT_REGISTER_SELECT_TBL"  number="28"
[4]       type="PRESENT_REGISTER_SELECT_RCD" label="Present Register Selection Table" role="CONTROL">
[5]       <description><b>PRESENT_REGISTER_SELECT_TBL</b> (Table 27) provides selections to sources of
[6]            present demands and present values. These indices point towards array elements in
[7]            <b>SOURCES_TBL</b> (Table 16) when Table 16 is present. The values selected are available in
[8]            <b>PRESENT_REGISTER_DATA_TBL</b> (Table 28). Table 27 is used to select sources for Table
[9]            28, which is continually updated.
[10]      </description>
[11]      <packedRecord name="PRESENT_REGISTER_SELECT_RCD">
[12]        <array name="PRESENT_DEMANDS_SELECT" type=" SOURCE_SELECT_RCD"
[13]          dimension="ACT_REGS_TBL.NBR_PRESENT_DEMANDS">
[14]          <description>Array that contains data source selectors for quantities used to measure present demand.
[15]            The present demand values shall be reported at the same index position in
[16]            <b>PRESENT_REGISTER_DATA_TBL. PRESENT_DEMAND</b> (Table 28).
[17]          </description>
[18]        </array>
[19]        <array name="PRESENT_VALUE_SELECT" type="SOURCE_SELECT_RCD"
[20]          dimension="ACT_REGS_TBL.NBR_PRESENT_VALUES">
[21]          <description>Array that contains data source selectors for quantities used to measure present value. The
[22]            present values shall be reported at the same index position in <b>PRESENT_REGISTER_DATA_TBL.
[23]            PRESENT_VALUES</b> (Table 28).
[24]          </description>
[25]        </array>
[26]      </packedRecord>
[27] </table>
[28]
[29] …
[30]
```

The CANDIDATE DOCUMENT exhibits the following errors:

- Lines 1, 3, 12, 22: Table number found `28`, should be `27`,
- Line 15: Extra “S” in element name. Element name should be `PRESENT_DEMAND_SELECT`, found `PRESENT_DEMANDS_SELECT`.
- Lines 16, 19: Missing “STD.” scope prefix to type `SOURCE_SELECT_RCD;`. Should be `STD.SOURCE_SELECT_RCD;`.

5.2 Exercise 2.

5.2.1 Extracted Section from the Referenced Standard Document Form

9.1.6 Table 05 Device Identification Table

Table 05 Data Description

DEVICE_IDENT_TBL (Table 05) This table provides the unique identifier for the device as specified by the user.

Global Default Table Property Overrides: Role="CONTROL"

Table 05 Type Definitions

```
TYPE IDENT_RCD = PACKED RECORD
        IF GEN_CONFIG_TBL.ID_FORM != 0 THEN
                IDENTIFICATION   : BCD(10);
        ELSE
                IDENTIFICATION   : STRING(20);
        END;
END;

TABLE 5 DEVICE_IDENT_TBL = IDENT_RCD;
```

Table 05 Element Descriptions

Identifier	Value	Definition
IDENT_RCD		
IDENTIFICATION		String of 20 characters or 20 BCD digits used to uniquely identify the device. This Element is the same as **UTIL_INFO_TBL.DEVICE_ID** when they both are available.

5.2.2 Extracted Section from the CANDIDATE DOCUMENT to Compare

[1] **9.1.6 Table 05 Device Identification Table**
[2]
[3] **Table 05 Data Description**
[4]
[5] **DEVICE_IDENT_TBL** (Table 05) This table provides the unique identifier for the device as specified by the user.
[6]
[7] **Global Default Table Property Overrides:** Role="DATA"
[8]
[9] **Table 05 Type Definitions**
[10]
[11] **TYPE IDENT = PACKED RECORD**
[12] **IF GEN_CONFIG_TBL.ID_FORM != 0 THEN**
[13] **IDENTIFICATION : BCD[20];**
[14] **ELSE;**
[15] **IDENTIFICATION : STRING[20];**
[16] **END;**
[17] **END;**
[18]
[19] **TABLE 5 DEVICE_IDENT_TBL = IDENT;**
[20]
[21] **Table 05 Element Descriptions**
[22]

[23] Identifier	Value	Definition
[24]		
[25] **IDENT**		
[26] **IDENTIFICATION**		String of 20 characters or 20 BCD digits used to uniquely identify
[27]		the device. This Element is the same as
[28]		**UTIL_INFO_TBL.DEVICE_ID** when they both are available.
[29]		
[30] **IDENTIFICATION**		String of 20 characters or 20 BCD digits used to uniquely identify
[31]		the device. This Element is the same as
[32]		**UTIL_INFO_TBL.DEVICE_ID** when they both are available.

[33]
[34]

5.2.3 Section from the CANDIDATE TDL/XML Form used to Produce the CANDIDATE DOCUMENT

```
[31]  …
[32]
[33]  <table name="DEVICE_IDENT_TBL" number="5"
[34]        type="IDENT" role="DATA" label="Device Identification Table">
[35]        <description><b>DEVICE_IDENT_TBL</b> (Table 05) This table provides the unique identifier for
[36]           the device as specified by the user.
[37]        </description>
[38]        <packedRecord name="IDENT ">
[39]           <if condition="GEN_CONFIG_TBL.ID_FORM != 0">
[40]              <then>
[41]                 <element name="IDENTIFICATION" type="BCD" length="20">
[42]                    <description>String of 20 characters or 20 BCD digits used to uniquely identify
[43]                       the device. This Element is the same as <b>UTIL_INFO_TBL.DEVICE_ID</b> when
[44]                       they both are available.
[45]                    </description>
[46]                 </element>
[47]              </then>
[48]              <else>
[49]                 <element name="IDENTIFICATION" type="STRING" length="20">
[50]                    <description>String of 20 characters or 20 BCD digits used to uniquely identify
[51]                       the device. This Element is the same as <b>UTIL_INFO_TBL.DEVICE_ID</b> when
[52]                       they both are available.
[53]                    </description>
[54]                 </element>
[55]              </else>
[56]              ;
[57]           </if>
[58]        </packedRecord>
[59]  </table>
[60]  …
```

5.2.4 Solution – Issues to be identified and reported by the ACCREDITOR

9.1.6 Table 05 Device Identification Table

Table 05 Data Description

DEVICE_IDENT_TBL (Table 05) This table provides the unique identifier for the device as specified by the user.

Global Default Table Property Overrides: Role="DATA"

Table 05 Type Definitions

```
TYPE IDENT = PACKED RECORD
        IF GEN_CONFIG_TBL.ID_FORM != 0 THEN
                IDENTIFICATION   : BCD[20];
        ELSE;
                IDENTIFICATION   : STRING[20];
        END;
END;

TABLE 5 DEVICE_IDENT_TBL = IDENT;
```

Table 05 Element Descriptions

Identifier	Value	Definition
IDENT		
IDENTIFICATION		String of 20 characters or 20 BCD digits used to uniquely identify the device. This Element is the same as **UTIL_INFO_TBL.DEVICE_ID** when they both are available.
IDENTIFICATION		String of 20 characters or 20 BCD digits used to uniquely identify the device. This Element is the same as **UTIL_INFO_TBL.DEVICE_ID** when they both are available.

5.2.5 Solution – Section from the CANDIDATE TDL/XML Form used to Produce the CANDIDATE DOCUMENT

```
[1]  …
[2]
[3]  <table name="DEVICE_IDENT_TBL" number="5"
[4]       type="IDENT" role="DATA" label="Device Identification Table">
[5]       <description><b>DEVICE_IDENT_TBL</b> (Table 05) This table provides the unique identifier for
[6]          the device as specified by the user.
[7]       </description>
[8]       <packedRecord name="IDENT ">
[9]          <if condition="GEN_CONFIG_TBL.ID_FORM != 0">
[10]            <then>
[11]               <element name="IDENTIFICATION" type="BCD" length="20">
[12]                  <description>String of 20 characters or 20 BCD digits used to uniquely identify
[13]                     the device. This Element is the same as <b>UTIL_INFO_TBL.DEVICE_ID</b> when
[14]                     they both are available.
[15]                  </description>
[16]               </element>
[17]            </then>
[18]            <else>
[19]               <element name="IDENTIFICATION" type="STRING" length="20">
[20]                  <description>String of 20 characters or 20 BCD digits used to uniquely identify
[21]                     the device. This Element is the same as <b>UTIL_INFO_TBL.DEVICE_ID</b> when
[22]                     they both are available.
[23]                  </description>
[24]               </element>
[25]            </else>
[26]            ;
[27]         </if>
[28]      </packedRecord>
[29] </table>
[30] …
```

The CANDIDATE DOCUMENT exhibits the following errors:

- Line 7: Role should be “CONTROL”.
- Lines 8, 19, 25: Missing packed record name suffix “`_RCD`”. Packed record name should be `IDENT_RCD`.
- Line 13: Incorrect dimension “`20`”. Should be `10`.
- Line 13: Incorrect array dimension delimiters “[” and “]”. Should be `BCD(10)`
- Line 14: Extra “;” after “`ELSE`”.
- Line 15: Incorrect array dimension delimiters “[” and “]”. Should be `STRING(20)`

Note: The duplication of the description of “`IDENTIFICATION`” although esthetically displeasing is not an error. The referenced Standard Document Form syntax calls for an “IF” clause and an “ELSE” clause that have identical descriptions for each instance of the element “`IDENTIFICATION`”. Although the XML/TDL Form syntax provides the facility for the suppression of such documentation, the CANDIDATE TDL/XML Form did not exercise this capability.

Note: The syntax section utilizes `BCD(10)` and `STRING(10)`, while the description states “String of 20 characters or 20 BCD digits…”, because .one BCD element contains one octet, which can hold two BCD digits (nibbles).

5.3 Exercise 3.

5.3.1 Extracted Section from the Referenced Standard Document Form

```
[1]   9.1.8      Table 07 Procedure Initiate Table
[2]
[3]   …
[4]
[5]   Table 07 Type Definitions
[6]
[7]   TYPE STD_PARM_RCD = PACKED RECORD
[8]   …
[9]
[10]          IF PROC.TABLE_PROC_NBR == REMOTE_RESET_PROC THEN
[11]                PROC_DATA_9 : REMOTE_RESET_PROC.PARM_DATA_RCD;
[12]          END;
[13]  …
[14]
[15]  END;
[16]
[17]  …
```

```
[18]  9.1.9      Table 08 Procedure Response Table
[19]
[20]  …
[21]
[22]  Table 08 Type Definitions
[23]
[24]          TYPE STD_RESP_RCD = PACKED RECORD
[25]  …
[26]
[27]                IF PROC.TABLE_PROC_NBR == REMOTE_RESET_PROC THEN
[28]              PROC_DATA_9 : REMOTE_RESET_PROC.RESP_DATA_RCD;
[29]                END;
[30]  …
[31]
[32]  END;
[33]
[34]  …
```

```
[35]  9.1.10.10 Procedure 09 Remote Reset
[36]
[37]  Procedure 09 Data Description
[38]
[39]          When invoked, this procedure attempts to perform the specified combination of resets.  The following res
[40]          are supported: Self-read, Demand Reset, and Season Change.
[41]
[42]  Procedure 09 Type Definitions
[43]
[44]          TYPE ACTION_FLAG_BFLD = BIT FIELD OF UINT8
[45]                DEMAND_RESET_FLAG     : BOOL(0);
[46]                SELF_READ_FLAG        : BOOL(1);
[47]                SEASON_CHANGE_FLAG    : BOOL(2);
[48]                NEW_SEASON            : UINT(3..6);
[49]                FILLER                : FILL(7..7);
[50]          END;
[51]
[52]          TYPE PARM_DATA_RCD = PACKED RECORD
[53]                ACTION_FLAG           : ACTION_FLAG_BFLD;
[54]          END;
```

```
[55]
[56]        TYPE RESP_DATA_RCD = PACKED RECORD
[57]              SUCCESS_FLAG          : ACTION_RESPONSE_FLAG_BFLD;
[58]        END;
[59]
[60]  PROCEDURE 9 REMOTE_RESET_PROC
[61]        REQUEST = PARM_DATA_RCD
[62]        RESPONSE = RESP_DATA_RCD;
[63]
[64]        Procedure 09 Element Descriptions
[65]
```

[66] Identifier	Value	Definition
[67]		
[68] **ACTION_FLAG_BFLD**		
[69] **DEMAND_RESET_FLAG**	FALSE	Do not perform a demand reset.
[70]	TRUE	Perform a demand reset.
[71]		
[72] **SELF_READ_FLAG**	FALSE	Do not perform a Self-read.
[73]	TRUE	Perform a Self-read.
[74]		
[75] **SEASON_CHANGE_FLAG**	FALSE	Do not perform a change to the new sea
[76]		specified.
[77]	TRUE	Perform a change to the new season specified
[78]		
[79] **NEW_SEASON**	0..15	Season to change to if **SEASON_CHANGE** b
[80]		set.
[81] **PARM_DATA_RCD**		
[82] **ACTION_FLAG**		Bit mask denotes the reset actions to perform.
[83]		
[84] **ACTION_RESPONSE_FLAG_BFLD**		**Redefines: ACTION_FLAG_BFLD**.
[85]		
[86] **DEMAND_RESET_FLAG**	FALSE	Did not perform a demand reset.
[87]	TRUE	Performed a demand reset.
[88]		
[89] **SELF_READ_FLAG**	FALSE	Did not perform a Self-read.
[90]	TRUE	Performed a Self-read.
[91]		
[92] **SEASON_CHANGE_FLAG**	FALSE	Did not perform a change to the new sea
[93]		specified.
[94]	TRUE	Performed a change to the new season specifi
[95]		
[96] **NEW_SEASON**	0..15	The season number the End Device changed
[97]		**SEASON_CHANGE** bit was set.
[98] **RESP_DATA_RCD**		
[99] **SUCCESS_FLAG**		Indicates which "resets" were successful by
[100]		setting corresponding bits.
[101]		
[102]		

5.3.2 Extracted Section from the CANDIDATE DOCUMENT to Compare

[1] [2] [3] [4] [5] [6] [7] [8] [9] [10] [11] [12] [13] [14] [15] [16] [17]	

[18] [19] [20] [21] [22] [23] [24] [25] [26] [27] [28] [29] [30] [31] [32] [33] [34]	

[35]	**9.1.10.10 Procedure 09 Remote Reset**
[36]	
[37]	**Procedure 09 Data Description**
[38]	
[39]	When invoked, this procedure executes the specified combination of resets. The following resets
[40]	supported: Self-read, Demand Reset, and Season Change.
[41]	
[42]	**Procedure 09 Type Definitions**
[43]	
[44]	**TYPE ACTION_FLAG_BFLD = BIT FIELD OF UINT8**
[45]	**DEMAND_RESET_FLAG : BOOL(0)**;
[46]	**SELF_READ_FLAG : BOOL(1);**
[47]	**SEASON_CHANGE_FLAG : BOOL(2);**
[48]	**NEW_SEASON : UINT(3..6);**
[49]	
[50]	**END;**
[51]	
[52]	**TYPE PARM_DATA_RCD = PACKED RECORD**
[53]	**ACTION_FLAG : ACTION_FLAG_BFLD**;
[54]	**END;**
[55]	
[56]	**TYPE RESP_DATA_RCD = PACKED RECORD**
[57]	**SUCCESS_FLAG : ACTION_RESPONSE_FLAG_BFLD**;
[58]	**END;**

PROCEDURE 9 REMOTE_RESET_
PROC REQUEST = PARM_DATA_RCD RESPONSE = RESP_DATA_RCD;

Procedure 09 Element Descriptions

Identifier	Value	Definition
ACTION_FLAG_BFLD		
DEMAND_RESET_FLAG	TRUE	Do not perform a demand reset.
	FALSE	Perform a demand reset.
SELF_READ_FLAG	FALSE	Do not perform a Self-read.
	TRUE	Perform a Self-read.
SEASON_CHANGE_FLAG	FALSE	Do not perform a change to the new sea specified.
	TRUE	Perform a change to the new season specified
NEW_SEASON	0..15	Season to change to if **SEASON_CHANGE** b set.
PARM_DATA_RCD		
ACTION_FLAG		Bit mask denotes the reset actions to perform.
ACTION_RESPONSE_FLAG_BFLD		
DEMAND_RESET_FLAG	FALSE	Did not perform a demand reset.
	TRUE	Performed a demand reset.
SELF_READ_FLAG	FALSE	Did not perform a Self-read.
	TRUE	Performed a Self-read.
SEASON_CHANGE_FLAG	FALSE	Did not perform a change to the new sea specified.
	TRUE	Performed a change to the new season specifi
NEW_SEASON	0..255	The season number the End Device changed **SEASON_CHANGE** bit was set.
RESP_DATA_RCD		
SUCCESS_FLAG		Indicates which "resets" were successful by setting corresponding bits.

5.3.3 Section from the CANDIDATE TDL/XML Form used to Produce the CANDIDATE DOCUMENT

```
[1]  …
[2]
[3]  <procedure name="REMOTE_RESET_PROC" number="9"  label="Remote Reset">
[4]      <description>When invoked, this procedure executes the specified combination of resets. The following resets
[5]          are supported: Self Read, Demand Reset, and Season Change.
[6]      </description>
[7]      <bitField name="ACTION_FLAG_BFLD" type="UINT8">
[8]        <subElement name="DEMAND_RESET_FLAG" type="BOOL" startBitInclusive="0">
[9]          <enumerator>
[10]           <enum value="false" text="Do not perform a demand reset." />
[11]           <enum value="true" text="Perform a demand reset." />
[12]         </enumerator>
[13]       </subElement>
[14]       <subElement name="SELF_READ_FLAG" type="BOOL" startBitInclusive="1">
[15]         <enumerator>
[16]           <enum value="false" text="Do not perform a Self-read." />
[17]           <enum value="true" text="Perform a Self-read." />
[18]         </enumerator>
[19]       </subElement>
[20]       <subElement name="SEASON_CHANGE_FLAG" type="BOOL" startBitInclusive="2">
[21]         <enumerator>
[22]           <enum value="false" text="Do not perform a change to the new season specified." />
[23]           <enum value="true" text="Perform a change to the new season specified." />
[24]         </enumerator>
[25]       </subElement>
[26]       <subElement name="NEW_SEASON" type="UINT" startBitInclusive="3" endBitInclusive="6">
[27]         <description>Season to change to if <b>SEASON_CHANGE</b> bit is set.</description>
[28]       </subElement>
[29]
[30]     </bitField>
[31]     <packedRecord name="PARM_DATA_RCD">
[32]       <element name="ACTION_FLAG" type="ACTION_FLAG_BFLD">
[33]         <description>Bit mask denotes the reset actions to perform.</description>
[34]       </element>
[35]     </packedRecord>
[36]     <bitField name="ACTION_RESPONSE_FLAG_BFLD" type="UINT8">
[37]       <subElement name="DEMAND_RESET_FLAG" type="BOOL" startBitInclusive="0">
[38]         <enumerator>
[39]           <enum value="true" text="Did not perform a demand reset." />
[40]           <enum value="false" text="Performed a demand reset." />
[41]         </enumerator>
[42]       </subElement>
[43]       <subElement name="SELF_READ_FLAG" type="BOOL" startBitInclusive="1">
[44]         <enumerator>
[45]           <enum value="false" text="Did not perform a Self-read." />
[46]           <enum value="true" text="Performed a Self-read." />
[47]         </enumerator>
[48]       </subElement>
[49]       <subElement name="SEASON_CHANGE_FLAG" type="BOOL" startBitInclusive="2">
[50]         <enumerator>
[51]           <enum value="false" text="Did not perform a change to the new season specified." />
[52]           <enum value="true" text="Performed a change to the new season specified." />
[53]         </enumerator>
[54]       </subElement>
[55]       <subElement name="NEW_SEASON" type="UINT8">
[56]         <description>The season number the End Device changed to if <b>SEASON_CHANGE</b> bit was
[57]           set.
[58]         </description>
[59]       </subElement>
[60]       <subElement name="FILLER" type="FILL" startBitInclusive="7" />
[61]     </bitField>
[62]     <packedRecord name="RESP_DATA_RCD">
[63]       <element name="SUCCESS_FLAG" type="ACTION_RESPONSE_FLAG_BFLD">
```

```
[64]                <description>Indicates which &#8220;resets&#8221; were successful by setting
[65]                  corresponding bits.</description>
[66]              </element>
[67]            </packedRecord>
[68]            <extend type="PARM_DATA_RCD" target="PROC_INITIATE_TBL.STD_PARM_RCD" />
[69]            <extend type="RESP_DATA_RCD" target="PROC_RESPONSE_TBL.STD_RESP_RCD" />
[70]        </procedure>
[71]        …
```

5.3.4 Solution – Issues to be identified and reported by the ACCREDITOR

[1]	
[2]	
[3]	
[4]	
[5]	
[6]	
[7]	
[8]	
[9]	
[10]	
[11]	
[12]	
[13]	
[14]	
[15]	
[16]	
[17]	

[18]	
[19]	
[20]	
[21]	
[22]	
[23]	
[24]	
[25]	
[26]	
[27]	
[28]	
[29]	
[30]	
[31]	
[32]	
[33]	
[34]	

[35]	**9 9.1.10.10 Procedure 09 Remote Reset**
[36]	
[37]	**Procedure 09 Data Description**
[38]	
[39]	When invoked, this procedure executes the specified combination of resets. The following resets are supported:
[40]	Self-read, Demand Reset, and Season Change.
[41]	
[42]	**Procedure 09 Type Definitions**
[43]	
[44]	**TYPE ACTION_FLAG_BFLD = BIT FIELD OF UINT8**
[45]	**DEMAND_RESET_FLAG : BOOL(0)**;
[46]	**SELF_READ_FLAG : BOOL(1);**
[47]	**SEASON_CHANGE_FLAG : BOOL(2);**
[48]	**NEW_SEASON : UINT8;**
[49]	
[50]	**END;**
[51]	
[52]	**TYPE PARM_DATA_RCD = PACKED RECORD**
[53]	**ACTION_FLAG : ACTION_FLAG_BFLD**;
[54]	**END;**
[55]	
[56]	**TYPE RESP_DATA_RCD = PACKED RECORD**
[57]	**SUCCESS_FLAG : ACTION_RESPONSE_FLAG_BFLD**;
[58]	**END;**

PROCEDURE 9 REMOTE_RESET_
PROC REQUEST = PARM_DATA_RCD RESPONSE = RESP_DATA_RCD;

Procedure 09 Element Descriptions

Identifier	Value	Definition
ACTION_FLAG_BFLD		
DEMAND_RESET_FLAG	TRUE	Do not perform a demand reset.
	FALSE	Perform a demand reset.
SELF_READ_FLAG	FALSE	Do not perform a Self-read.
	TRUE	Perform a Self-read.
SEASON_CHANGE_FLAG	FALSE	Do not perform a change to the new season specified.
	TRUE	Perform a change to the new season specified.
NEW_SEASON	0..15	Season to change to if **SEASON_CHANGE** bit is set.
PARM_DATA_RCD		
ACTION_FLAG		Bit mask denotes the reset actions to perform.
ACTION_RESPONSE_FLAG_BFLD		
DEMAND_RESET_FLAG	FALSE	Did not perform a demand reset.
	TRUE	Performed a demand reset.
SELF_READ_FLAG	FALSE	Did not perform a Self-read.
	TRUE	Performed a Self-read.
SEASON_CHANGE_FLAG	FALSE	Did not perform a change to the new season specified.
	TRUE	Performed a change to the new season specified.
NEW_SEASON	0..255	The season number the End Device changed to if **SEASON_CHANGE** bit was set.
RESP_DATA_RCD		
SUCCESS_FLAG		Indicates which "resets" were successful by setting corresponding bits.

5.3.5 Solution – Section from the CANDIDATE TDL/XML Form used to Produce the CANDIDATE DOCUMENT

```
[1]  …
[2]
[3]  <procedure name="REMOTE_RESET_PROC" number="9"  label="Remote Reset">
[4]      <description>When invoked, this procedure executes the specified combination of resets. The following resets
[5]          are supported: Self Read, Demand Reset, and Season Change.
[6]      </description>
[7]      <bitField name="ACTION_FLAG_BFLD" type="UINT8">
[8]        <subElement name="DEMAND_RESET_FLAG" type="BOOL" startBitInclusive="0">
[9]          <enumerator>
[10]           <enum value="false" text="Do not perform a demand reset." />
[11]           <enum value="true" text="Perform a demand reset." />
[12]         </enumerator>
[13]       </subElement>
[14]       <subElement name="SELF_READ_FLAG" type="BOOL" startBitInclusive="1">
[15]         <enumerator>
[16]           <enum value="false" text="Do not perform a Self-read." />
[17]           <enum value="true" text="Perform a Self-read." />
[18]         </enumerator>
[19]       </subElement>
[20]       <subElement name="SEASON_CHANGE_FLAG" type="BOOL" startBitInclusive="2">
[21]         <enumerator>
[22]           <enum value="false" text="Do not perform a change to the new season specified." />
[23]           <enum value="true" text="Perform a change to the new season specified." />
[24]         </enumerator>
[25]       </subElement>
[26]       <subElement name="NEW_SEASON" type="UINT" startBitInclusive="3" endBitInclusive="6">
[27]         <description>Season to change to if <b>SEASON_CHANGE</b> bit is set.</description>
[28]       </subElement>
[29]
[30]     </bitField>
[31]     <packedRecord name="PARM_DATA_RCD">
[32]       <element name="ACTION_FLAG" type="ACTION_FLAG_BFLD">
[33]         <description>Bit mask denotes the reset actions to perform.</description>
[34]       </element>
[35]     </packedRecord>
[36]     <bitField name="ACTION_RESPONSE_FLAG_BFLD" type="UINT8">
[37]       <subElement name="DEMAND_RESET_FLAG" type="BOOL" startBitInclusive="0">
[38]         <enumerator>
[39]           <enum value="true" text="Did not perform a demand reset." />
[40]           <enum value="false" text="Performed a demand reset." />
[41]         </enumerator>
[42]       </subElement>
[43]       <subElement name="SELF_READ_FLAG" type="BOOL" startBitInclusive="1">
[44]         <enumerator>
[45]           <enum value="false" text="Did not perform a Self-read." />
[46]           <enum value="true" text="Performed a Self-read." />
[47]         </enumerator>
[48]       </subElement>
[49]       <subElement name="SEASON_CHANGE_FLAG" type="BOOL" startBitInclusive="2">
[50]         <enumerator>
[51]           <enum value="false" text="Did not perform a change to the new season specified." />
[52]           <enum value="true" text="Performed a change to the new season specified." />
[53]         </enumerator>
[54]       </subElement>
[55]       <subElement name="NEW_SEASON" type="UINT8">
[56]         <description>The season number the End Device changed to if <b>SEASON_CHANGE</b> bit was
[57]           set.
[58]         </description>
[59]       </subElement>
[60]       <subElement name="FILLER" type="FILL" startBitInclusive="7" />
[61]     </bitField>
[62]     <packedRecord name="RESP_DATA_RCD">
[63]       <element name="SUCCESS_FLAG" type="ACTION_RESPONSE_FLAG_BFLD">
```

```
[64]              <description>Indicates which &#8220;resets&#8221; were successful by setting
[65]                corresponding bits.</description>
[66]            </element>
[67]          </packedRecord>
[68]          <extend type="PARM_DATA_RCD" target="PROC_INITIATE_TBL.STD_PARM_RCD" />
[69]          <extend type="RESP_DATA_RCD" target="PROC_RESPONSE_TBL.STD_RESP_RCD" />
[70]    </procedure>
[71]    ...
```

The CANDIDATE DOCUMENT exhibits the following errors:

- Lines 1 through 24: Missing auto generated syntax in for inclusion in Table 7 and Table 8.
- Line 39: Standard text is “attempts to perform” found “executes”.
- Line 48: Incorrect definition of sub-element “`NEW_SEASON`”, found `UINT8`, expecting `UINT(3..6)`.
- Line 49: Missing statement `FILLER: FILL(7..7);` .
- Line 68: Incorrect values for of TRUE/FALSE in description of sub element `DEMAND_RESET_FLAG`.
- Line 85: Missing redefinition clause of type “`ACTION_RESPONSE_FLAG_BFLD`”. Expected “`Redefines: ACTION_FLAG_BFLD`”.
- Line 99: Incorrect description of sub-element “`NEW_SEASON`”, found `0..255`, expecting `0..15`.

Note:.The missing Lines 1 through 24 in the auto generated syntax of the DOCUMENT do not necessarily represent a fatal error, if the TXL/XML form is correctly encoded. However, the text in Tables 7 and 8 provide insight into CANDIDATE’s understanding of the production rules for scoped elements. TDL/XML Form procedure types are defined in the scope of the local scope of the procedure. They must be preserved when the same types are referenced from the remote scope of Tables 7 and 8, as shown in the Referenced Standard Document Form. This defect, may indicative of a problem, the problem may be only related to the production of the DOCUMENT and not the TDL/XML Form.

Note:.The missing “Redefine:” clause in the description is an important indication that the TDL/XML Form is incorrectly formulated. One should be on the lookout for terms such as “Redefine:”, “Replace:”, “Override:”, because they represent restrictions on syntax that TDL compilers need to be aware of.

5.4 Exercise 4.

5.4.1 Extracted Section from the Referenced Standard Document Form

9.5 Decade 4: Security Tables

Decade 4 Name
SECURITY_DEC

Decade 4 Data Description

The security tables provide holding areas for the placement of End Device passwords and Encryption / Authentication keys used to establish group access permissions. Access permissions are used to limit table read or write access and procedure execute permissions to groups of users based on the interpretation of the password and encryption/authentication fields. The exact means for granting access are not defined by this standard.

9.5.1 Table 40 Security Dimension Limits Table

Table 40 Data Description

DIM_SECURITY_LIMITING_TBL (Table 40) defines the maximum number of passwords and security access level entries supported by the End Device.

Global Default Table Property Overrides: Role="LIMITING", Accessibility="READONLY"

Table 40 Type Definitions

```
TYPE SECURITY_RCD = PACKED RECORD
        NBR_PASSWORDS       : UINT8;
        PASSWORD_LEN        : UINT8;
        NBR_KEYS            : UINT8;
        KEY_LEN             : UINT8;
        NBR_PERM_USED       : UINT16;
END;

TABLE 40 DIM_SECURITY_LIMITING_TBL = SECURITY_RCD;
```

Table 40 Element Descriptions

Identifier	Value	Definition
SECURITY_RCD		
NBR_PASSWORDS	0	Access levels cannot be established using Standard password mechanism.
	1..255	Maximum number of passwords this End Dev is capable of supporting.
PASSWORD_LEN	0	Reserved.
	1..20	Maximum length of passwords in octets this I Device is capable of supporting.
	21..255	Reserved.
NBR_KEYS	0..255	Maximum number of keys authentication/encryption this End Device capable of supporting.
KEY_LEN	0..255	Maximum length of keys in octets this End Dev is capable of supporting.
NBR_PERM_USED	0..65535	Maximum number of user-defined security access entries this End Device supports in addition to those defined through default access permissions.

5.4.2 Extracted Section from the CANDIDATE DOCUMENT to Compare

9.5 Decade 40: Security Tables

Decade 4 Name
SECURITY_DEC

Decade 4 Data Description

The security tables provide holding areas for the placement of End Device passwords and Encryption / Authentication keys used to establish group access permissions. Access permissions are used to limit table read or write access and procedure execute permissions to groups of users based on the interpretation of the password and encryption/authentication fields. The exact means for granting access are not defined by this standard.

9.5.1 Table 40 Security Dimension Limits Table

Table 40 Data Description

DIM_SECURITY_LIMITING_TBL (Table 40) defines the maximum number of passwords and security access level entries supported by the End Device.

Global Default Table Property Overrides: Role="LIMITING", Accessibility="READONLY"

Table 40 Type Definitions

```
SECURITY_RCD = PACKED RECORD
        NBR_PASSWORDS          : UINT8;
        PASSWORD_ LEN          : UINT8;
        NBR_KEYS               : UINT8;
        KEY_LEN                : UINT8;
        NBR_PERM_USED          : UINT8;
END;

TABLE 40 DIM_SECURITY_LIMITING_TBL = SECURITY_RCD;
```

Table 40 Element Descriptions

Identifier	Value	Definition
SECURITY_RCD		
NBR_PASSWORDS	0	Access levels can be established using Standard password mechanism.
	1..255	Maximum number of passwords this End De is capable of supporting.
PASSWORD_LEN		
	1..20	Maximum length of passwords in octets this Device is capable of supporting.
	21..255	Reserved.
NBR_KEYS	1..256	Maximum number of keys authentication/encryption this End Device capable of supporting.
KEY_LEN	0..255	Maximum length of keys in octets this End De is capable of supporting.
NBR_PERM_USED	0..65535	Maximum number of user-defined security access entries this End Device supports in addition to those defined through default access permissions.

5.4.3 Section from the CANDIDATE TDL/XML Form used to Produce the CANDIDATE DOCUMENT

```
[1]  …
[2]
[3]  <decade  number="40" label="Security Tables">
[4]       <description>
[5]          <b> Decade 4 Name<br />SECURITY_DEC</b>
[6]          <p>The security tables provide holding areas for the placement of End Device passwords and
[7]               Encryption/Authentication keys used to establish group access permissions. Access permissions are
[8]               used to limit table read or write access and procedure execute permissions to groups of users based on
[9]               the interpretation of the password and encryption/authentication fields. The exact means for granting
[10]              access are not defined by this standard.</p>
[11]      </description>
[12]
[13]      <table name="DIM_SECURITY_LIMITING_TBL" number="40" type="SECURITY_RCD"
[14]          label="Security Dimension Limits Table >
[15]         <description>
[16]            <b>Global Default Table Property Overrides:</b> Role=“LIMITING”, Accessibility=“READONLY”</b>
[17]            <p><b>DIM_SECURITY_LIMITING_TBL</b> (Table 40) defines the maximum number of
[18]            passwords and security access level entries supported by the End Device.</p>
[19]         </description>
[20]         <packedRecord name="SECURITY_RCD">
[21]            <element name="NBR_PASSWORDS" type="UINT8">
[22]               <enumerator>
[23]                  <enum value="0"
[24]                  text="Access levels cannot be established using the Standard password mechanism." />
[25]                  <enum value="1" endValueInclusive="255"
[26]                     text="Maximum number of passwords this End Device is capable of supporting." />
[27]               </enumerator>
[28]            </element>
[29]            <element name="PASSWORD_  LEN" type="UINT8" min="0" max="20">
[30]               <enumerator>
[31]                  <enum value="1" endValueInclusive="20"
[32]                     text="Maximum length of passwords in octets this End Device is capable of supporting." />
[33]                  <enum value="21" endValueInclusive="255" text="Reserved." exclude="true" />
[34]               </enumerator>
[35]            </element>
[36]            <element name="NBR_KEYS" type="UINT8" min=”1” max=”256”>
[37]               <description>Maximum number of keys for authentication/encryption this End Device is capable of
[38]                  supporting.
[39]               </description>
[40]            </element>
[41]            <element name="KEY_LEN" type="UINT8">
[42]               <description>Maximum length of keys in octets this End Device is capable of supporting.</description>
[43]            </element>
[44]            <element name="NBR_PERM_USED" type="UINT8" max=”65535”>
[45]               <description>Maximum number of user-defined security access entries this End Device supports in
[46]                  addition to those defined through default access permissions.
[47]               </description>
[48]            </element>
[49]         </packedRecord>
[50]      </table>
[51]
[52]  …

      </decade>
      …
```

5.4.4 Solution – Issues to be identified and reported by the ACCREDITOR

9.5 Decade 40: Security Tables

Decade 4 Name
SECURITY_DEC

Decade 4 Data Description

The security tables provide holding areas for the placement of End Device passwords and Encryption / Authentication keys used to establish group access permissions. Access permissions are used to limit table read or write access and procedure execute permissions to groups of users based on the interpretation of the password and encryption/authentication fields. The exact means for granting access are not defined by this standard.

9.5.1 Table 40 Security Dimension Limits Table

Table 40 Data Description

DIM_SECURITY_LIMITING_TBL (Table 40) defines the maximum number of passwords and security access level entries supported by the End Device.

Global Default Table Property Overrides: Role="LIMITING", Accessibility="READONLY"

Table 40 Type Definitions

```
SECURITY_RCD = PACKED RECORD
        NBR_PASSWORDS          : UINT8;
        PASSWORD_ LEN          : UINT8;
        NBR_KEYS               : UINT8;
        KEY_LEN                : UINT8;
        NBR_PERM_USED          : UINT8;
END;

TABLE 40 DIM_SECURITY_LIMITING_TBL = SECURITY_RCD;
```

Table 40 Element Descriptions

Identifier	Value	Definition
SECURITY_RCD		
NBR_PASSWORDS	0	Access levels can be established using Standard password mechanism.
	1..255	Maximum number of passwords this End De is capable of supporting.
PASSWORD_LEN		
	1..20	Maximum length of passwords in octets this Device is capable of supporting.
	21..255	Reserved.
NBR_KEYS	1..256	Maximum number of keys authentication/encryption this End Device capable of supporting.
KEY_LEN	0..255	Maximum length of keys in octets this End De is capable of supporting.
NBR_PERM_USED	0..65535	Maximum number of user-defined security access entries this End Device supports in addition to those defined through default access permissions.

5.4.5 Solution – Section from the CANDIDATE TDL/XML Form used to Produce the CANDIDATE DOCUMENT

```
[1]   …
[2]
[3]   <decade  number="40" label="Security Tables">
[4]       <description>
[5]         <b> Decade 4 Name<br />SECURITY_DEC</b>
[6]         <p>The security tables provide holding areas for the placement of End Device passwords and
[7]               Encryption/Authentication keys used to establish group access permissions. Access permissions are
[8]               used to limit table read or write access and procedure execute permissions to groups of users based on
[9]               the interpretation of the password and encryption/authentication fields. The exact means for granting
[10]              access are not defined by this standard.</p>
[11]      </description>
[12]
[13]      <table name="DIM_SECURITY_LIMITING_TBL" number="40" type="SECURITY_RCD"
[14]          label="Security Dimension Limits Table >
[15]        <description>
[16]          <b>Global Default Table Property Overrides:</b> Role=“LIMITING”, Accessibility=“READONLY”</b>
[17]          <p><b>DIM_SECURITY_LIMITING_TBL</b> (Table 40) defines the maximum number of
[18]          passwords and security access level entries supported by the End Device.</p>
[19]        </description>
[20]        <packedRecord name="SECURITY_RCD">
[21]          <element name="NBR_PASSWORDS" type="UINT8">
[22]            <enumerator>
[23]              <enum value="0"
[24]              text="Access levels cannot be established using the Standard password mechanism." />
[25]              <enum value="1" endValueInclusive="255"
[26]                text="Maximum number of passwords this End Device is capable of supporting." />
[27]            </enumerator>
[28]          </element>
[29]          <element name="PASSWORD_ LEN" type="UINT8" min="0" max="20">
[30]            <enumerator>
[31]              <enum value="1" endValueInclusive="20"
[32]                text="Maximum length of passwords in octets this End Device is capable of supporting." />
[33]              <enum value="21" endValueInclusive="255" text="Reserved." exclude="true" />
[34]            </enumerator>
[35]          </element>
[36]          <element name="NBR_KEYS" type="UINT8" min=”1” max=”256”>
[37]            <description>Maximum number of keys for authentication/encryption this End Device is capable of
[38]              supporting.
[39]            </description>
[40]          </element>
[41]          <element name="KEY_LEN" type="UINT8">
[42]            <description>Maximum length of keys in octets this End Device is capable of supporting.</description>
[43]          </element>
[44]          <element name="NBR_PERM_USED" type="UINT8" max=”65535”>
[45]            <description>Maximum number of user-defined security access entries this End Device supports in
[46]              addition to those defined through default access permissions.
[47]            </description>
[48]          </element>
[49]        </packedRecord>
[50]      </table>
[51]
[52]  …

      </decade>
      …
```

The CANDIDATE DOCUMENT exhibits the following errors:

- Line 1: Incorrect Decade number. Found “40” should be 4, see next item below for correction.

- Line 4: Decade name identifier is presented as documentation only. Review of TDL/XML decade declaration exposes the description element text: "`<description><b> Decade 4 Name<br />SECURITY_DEC</b>…</description>`", expected content should be in decade introducer as shown below:

  ```
  <decade name="SECURITY_DEC" number="4" label="Security
  Tables">.
  ```

- Line 20: Table properties are presented as documentation only. Review of TDL/XML decade declaration exposes the description element text: "`<description>Global Default Table Property Overrides: Role="LIMITING", Accessibility="READONLY"… </description>`", expected content should be in table introducer as shown below:

  ```
  <table name="DIM_SECURITY_LIMITING_TBL" number="40"
  type="SECURITY_RCD" label="Security Dimension Limits Table"
  role="LIMITING" accessibility="READONLY">.
  ```

  Alternatively the role and the accessibility could be supplied using the "qualify" element, per Section I.2.27, "<qualify> Element (Child of <tdl>)", Ref: [1] and [2], as shown below:

  ```
  </qualify><table name="DIM_SECURITY_LIMITING_TBL"
  role="LIMITING" accessibility="READONLY" </table></qualify>
  ```

- Line 24: Missing keyword "`TYPE`" in definition of "`SECURITY_RCD`".
- Line 26: Extra space found in element name "`PASSWORD_  LEN`", should be `PASSWORD_LEN`.
- Line 29: Incorrect type definition of element "`NBR_PERM_USED`". Found "`UINT8`", expecting `UINT16`.
- Line 39: Incorrect transcription of description of "`NBR_PASSWORDS`". Expecting "`Access levels cannot be established`", found `Access levels can be established`...
- Line 39: Incorrect in-line enumeration of description of "`PASSWORD_LEN`". Missing:

  `0 Reserved`.

- Line 49: Incorrect in-line range of description of "`NBR_KEYS`". Found: "`1..256`", expecting `0..255`.

Note:.The discrepancies in lines 4 and 20 may not be discovered by a casual read of the DOCUMENT. However, targeted sampling the TDL/XML Form file, following the initial valuation will expose these issues. Extra confidence may be gained through the manipulation of select control fields of the DOCUMENT and review of the resulting DOCUMENT.

5.5 Exercise 5.

5.5.1 Extracted Section from the Referenced Standard Document Form

6.1 Character Set Selection

This selection is used to determine the encoding of characters that are used in the Tab
GEN_CONFIG_TBL.FORMAT_CONTROL_1.CHAR_FORMAT is the controlling selector for the encoding of **CH**
STRING, **FLOAT_CHAR6**, **FLOAT_CHAR12** and **FLOAT_CHAR21**.

TDL Type Definitions

{ *Enumerator* ***CHAR_FORMAT_ENUM*** **}**

TDL Element Descriptions

Identifier	Value	Definition
CHAR_FORMAT_ENUM		
	0	Reserved.
	1	ISO 7-bit coded character set for informa interchange, per ISO/IEC 646: 1991, a fixed-w encoding using 8 bits as its base unit for ler calculations and **UINT8** as the base unit transmission.
	2	ISO 8-bit coded character as per ISO 8859/1 ECMA-94 Latin 1 character set, a fixed-w encoding using 8 bits as its base unit for ler calculations and **UINT8** as the base unit transmission.
	3	UTF-8 as per *utf8*., variable-width encoding us 8 bits as its base unit for length calculations **UINT8** as the base unit for transmission. This t maximizes compatibility with **CHAR_FORMAT** and 2 as per Section 2.5, “Encoding Forms / U 8” and Section 15.9, “Specials“ of “The Unic Standard”, Version 4.0. In UTF-8 the byte or mark, BOM, corresponds to the byte seque [#xEF] [#xBB] [#xBF]. The BOM of UTF-8 shal permitted only when it can be accommodated the size of the element that is of type UT whose size is at least three octets, but it shal ignored (not produce displayable text) by processing application.
	4..7	Reserved.

5.5.2 Extracted Section from the CANDIDATE DOCUMENT to Compare

6.1 Character Set Selection

This selection is used to determine the encoding of characters that are used in the Tables.
GEN_CONFIG_TBL.FORMAT_CONTROL_1.CHAR_FORMAT is the controlling selector for the encoding of CHAR, STRING, FLOAT_CHAR6, FLOAT_CHAR12 and FLOAT_CHAR21.

TDL Type Definitions

{ *Enumerator CHAR_FORMAT_ENUM* }

TDL Element Descriptions

Identifier	Value	Definition
CHAR_FORMAT_ENUM		
	0	Reserved.
	1	ISO 7-bit coded character set for information interchange, per ISO/IEC 646: 1991, a fixed-width encoding using 8 bits as its base unit for length calculations and UINT8 as the base unit for transmission.
	2	ISO 8-bit coded character as per ISO 8859/1 or ECMA-94 Latin 1 character set, a fixed-width encoding using 8 bits as its base unit for length calculations and UINT8 as the base unit for transmission.
	3	UTF-8 as per *utf8.*, variable-width encoding using 8 bits as its base unit for length calculations and UINT8 as the base unit for transmission. This type maximizes compatibility with CHAR_FORMATs 1 and 2 as per Section 2.5, “Encoding Forms / UTF-8” and Section 15.9, “Specials“ of “The Unicode Standard”, Version 4.0. In UTF-8 the byte order mark, BOM, corresponds to the byte sequence [#xEF] [#xBB] [#xBF]. The BOM of UTF-8 shall be permitted only when it can be accommodated by the size of the element that is of type UTF-8 whose size is at least three octets, but it shall be ignored (not produce displayable text) by the processing application.
	4..7	Reserved.

5.5.3 Section from the CANDIDATE TDL/XML Form used to Produce the CANDIDATE DOCUMENT

```
[1]  …
[2]      <description>
[3]        <h2>Character Set Selection</h2>
[4]        <p>This selection is used to determine the encoding of characters that are used in the Tables.
[5]              <b>GEN_CONFIG_TBL.FORMAT_CONTROL_1.CHAR_FORMAT</b> is the controlling selector for
[6]              the encoding of <b>CHAR</b>, <b>STRING</b>, <b>FLOAT_CHAR6</b>, <b>FLOAT_CHAR12</b>
[7]              and <b>FLOAT_CHAR21</b>.
[8]        </p>
[9]      </description>
[10]     <enumerator name="CHAR_FORMAT_ENUM">
[11]       <description documentation="false">This selection is used to determine the encoding of characters that are
[12]                 used in the Tables. <b>GEN_CONFIG_TBL.FORMAT_CONTROL_1.CHAR_FORMAT</b> is the
[13]                 controlling selector for the encoding of <b>CHAR</b>, <b>STRING</b>,
[14]                 <b>FLOAT_CHAR6</b>, <b>FLOAT_CHAR12</b> and <b>FLOAT_CHAR21</b>
[15]       </description>
[16]         <enum value="0" text="Reserved." exclude="true"> </enum>
[17]         <enum value="1" text="ISO 7-bit coded character set for information interchange, per ISO/IEC 646:
[18]                 1991.">
[19]           <description>ISO 7-bit coded character set for information interchange, per ISO/IEC 646: 1991, a fixed-
[20]                 width encoding using an 8-bit as its base unit for length calculations and <b>UINT8</b> as the
[21]                 base unit for transmission
[22]           </description>
[23]         </enum>
[24]         <enum value="2" text="ISO 8-bit coded character as per ISO 8859/1 or ECMA-94 Latin 1 character set.">
[25]           <description>ISO 8-bit coded character as per ISO 8859/1 or ECMA-94 Latin 1 character set, a fixed-
[26]                 width encoding using an 8-bit as its base unit for length calculations and <b>UINT8</b> as the
[27]                 base unit for transmission.
[28]           </description>
[29]         </enum>
[30]         <enum value="3" text="UTF-8 as per utf8., variable-width encoding using an 8-bit as its base unit for
[31]                 length calculations and UINT8 as the base unit for transmission.">
[32]           <description>UTF-8 as per utf8., variable-width encoding using an 8-bit as its base unit for length
[33]                 calculations and <b>UINT8</b> as the base unit for transmission. This type maximizes
[34]                 compatibility with <b>CHAR_FORMAT</b>s 1 and 2 as per Section 2.5, &amp;#8220;Encoding Forms /
[35]                 UTF-8&amp;#8221; and Section 15.9, &amp;#8220;Specials&amp;#8221; of &amp;#8220;The Unicode
[36]                 Standard&amp;#8221;, Version 4.0. In UTF-8 the byte order mark, BOM, corresponds to the byte
[37]                 sequence [#xEF] [#xBB] [#xBF]. The BOM of UTF-8 shall be permitted only when it can be
[38]                 accommodated by the size of the element that is of type
[39]                                 UTF-8 whose size is at least three octets, but it shall be ignored (not produce
[40]                 displayable text) by the processing application.
[41]           </description>
[42]         </enum>
[43]         <enum value="4" endValueInclusive="7" text="Reserved." exclude="true"> </enum>
[44]       </enumerator>
[45] …
[46]
```

5.5.4 Solution – Issues to be identified and reported by the ACCREDITOR

6.1 Character Set Selection

This selection is used to determine the encoding of characters that are used in the Tables.
GEN_CONFIG_TBL.FORMAT_CONTROL_1.CHAR_FORMAT is the controlling selector for the encoding of CHAR, STRING, FLOAT_CHAR6, FLOAT_CHAR12 and FLOAT_CHAR21.

TDL Type Definitions

{ *Enumerator CHAR_FORMAT_ENUM* }

TDL Element Descriptions

Identifier	Value	Definition
CHAR_FORMAT_ENUM		
	0	Reserved.
	1	ISO 7-bit coded character set for information interchange, per ISO/IEC 646: 1991, a fixed-width encoding using 8 bits as its base unit for length calculations and UINT8 as the base unit for transmission.
	2	ISO 8-bit coded character as per ISO 8859/1 or ECMA-94 Latin 1 character set, a fixed-width encoding using 8 bits as its base unit for length calculations and UINT8 as the base unit for transmission.
	3	UTF-8 as per *utf8.*, variable-width encoding using 8 bits as its base unit for length calculations and UINT8 as the base unit for transmission. This type maximizes compatibility with CHAR_FORMATs 1 and 2 as per Section 2.5, “Encoding Forms / UTF-8” and Section 15.9, “Specials“ of “The Unicode Standard”, Version 4.0. In UTF-8 the byte order mark, BOM, corresponds to the byte sequence [#xEF] [#xBB] [#xBF]. The BOM of UTF-8 shall be permitted only when it can be accommodated by the size of the element that is of type UTF-8 whose size is at least three octets, but it shall be ignored (not produce displayable text) by the processing application.
	4..7	Reserved.

5.5.5 Solution – Section from the CANDIDATE TDL/XML Form used to Produce the CANDIDATE DOCUMENT

```
[1]  …
[2]      <description>
[3]        <h2>Character Set Selection</h2>
[4]        <p>This selection is used to determine the encoding of characters that are used in the Tables.
[5]              <b>GEN_CONFIG_TBL.FORMAT_CONTROL_1.CHAR_FORMAT</b> is the controlling selector for
[6]              the encoding of <b>CHAR</b>, <b>STRING</b>, <b>FLOAT_CHAR6</b>, <b>FLOAT_CHAR12</b>
[7]              and <b>FLOAT_CHAR21</b>.
[8]        </p>
[9]      </description>
[10]     <enumerator name="CHAR_FORMAT_ENUM">
[11]       <description documentation="false">This selection is used to determine the encoding of characters that are
[12]                 used in the Tables. <b>GEN_CONFIG_TBL.FORMAT_CONTROL_1.CHAR_FORMAT</b> is the
[13]                 controlling selector for the encoding of <b>CHAR</b>, <b>STRING</b>,
[14]                 <b>FLOAT_CHAR6</b>, <b>FLOAT_CHAR12</b> and <b>FLOAT_CHAR21</b>
[15]       </description>
[16]         <enum value="0" text="Reserved." exclude="true"> </enum>
[17]         <enum value="1" text="ISO 7-bit coded character set for information interchange, per ISO/IEC 646:
[18]                 1991.">
[19]           <description>ISO 7-bit coded character set for information interchange, per ISO/IEC 646: 1991, a fixed-
[20]                 width encoding using an 8-bit as its base unit for length calculations and <b>UINT8</b> as the
[21]                 base unit for transmission
[22]           </description>
[23]         </enum>
[24]         <enum value="2" text="ISO 8-bit coded character as per ISO 8859/1 or ECMA-94 Latin 1 character set.">
[25]           <description>ISO 8-bit coded character as per ISO 8859/1 or ECMA-94 Latin 1 character set, a fixed-
[26]                 width encoding using an 8-bit as its base unit for length calculations and <b>UINT8</b> as the
[27]                 base unit for transmission.
[28]           </description>
[29]         </enum>
[30]         <enum value="3" text="UTF-8 as per utf8., variable-width encoding using an 8-bit as its base unit for
[31]                 length calculations and UINT8 as the base unit for transmission.">
[32]           <description>UTF-8 as per utf8., variable-width encoding using an 8-bit as its base unit for length
[33]                 calculations and <b>UINT8</b> as the base unit for transmission. This type maximizes
[34]                 compatibility with <b>CHAR_FORMAT</b>s 1 and 2 as per Section 2.5, &amp;#8220;Encoding Forms /
[35]                 UTF-8&amp;#8221; and Section 15.9, &amp;#8220;Specials&amp;#8221; of &amp;#8220;The Unicode
[36]                 Standard&amp;#8221;, Version 4.0. In UTF-8 the byte order mark, BOM, corresponds to the byte
[37]                 sequence [#xEF] [#xBB] [#xBF]. The BOM of UTF-8 shall be permitted only when it can be
[38]                 accommodated by the size of the element that is of type
[39]                                 UTF-8 whose size is at least three octets, but it shall be ignored (not produce
[40]                 displayable text) by the processing application.
[41]           </description>
[42]         </enum>
[43]         <enum value="4" endValueInclusive="7" text="Reserved." exclude="true"> </enum>
[44]       </enumerator>
[45] …
[46]
```

The CANDIDATE DOCUMENT exhibits no errors

- Lines 9 and 15: The DOCUMENT does not produce documentation. This is not an error for the following reasons. (a) The documentation was produced on line 3, according to the standard; and (b) the documentation was intentionally suppressed in the TDL/XML File on line 11;
- Line 9: No syntax was produced for the enumerator, other than noting its presence as a comment in the syntax section as follows: `{ Enumerator CHAR_FORMAT_ENUM }`. This is consistent with the production rules of the ANSI C12.19 standard.

- Lines 18, 24 and 30: The TDL/XML Form document's "`enum`" elements contain both a "`text`" attribute and a "`description`" element (see TDL/XML Form lines 17, 24 and 30. When the "`description`" element is present, then it is not at error to suppress the output of the "`text`" attribute of an "`enum`" element. The "`text`" attribute is then interpreted a redundant brief description only. However, when the "`description`" element is absent then the "`text`" attribute SHOULD express the enumerator description accurately pre referenced Standard Document Form. See line 23 TDL/XML Form for an example.

5.6 Exercise 6.

5.6.1 Extracted Section from the Referenced Standard Document Form

9.8.3 Table 72 Events Identification Table

Table 72 Data Description

EVENTS_ID_TBL (Table 72) contains the events that are supported by the End Device. Any of these events may expressed in the History Log Data (Table 74) and the Event Log Data (Table 76).

Global Default Table Property Overrides: Role="CONTROL"

Table 72 Type Definitions

```
TYPE EVENTS_SUPPORTED_RCD = PACKED RECORD
        STD_EVENTS_SUPPORTED        : SET(ACT_LOG_TBL.NBR_STD_EVENTS);
        MFG_EVENTS_SUPPORTED        : SET(ACT_LOG_TBL.NBR_MFG_EVENTS);
END;

TABLE 72 EVENTS_ID_TBL = EVENTS_SUPPORTED_RCD;
```

Table 72 Element Descriptions

Identifier	Value	Definition
EVENTS_SUPPORTED_RCD		
STD_EVENTS_SUPPORTED		This set Element indicates which of the standard events are supported in the Event Log. See Annex B, "History & Event Log Codes", for standard event codes. Event codes are represented by bits 0 through (8 * **ACT_LOG_TBL.NBR_STD_EVENTS** - 1), with a one (1) representing a TRUE or implemented condition and a zero (0) representing a FALSE or not implemented condition.
MFG_EVENTS_SUPPORTED		This set Element indicates which of the manufacturer events are supported in the Event Log. Events are enabled by bits 0 through (8 * **ACT_LOG_TBL.NBR_MFG_EVENTS** - 1), with a one (1) representing a TRUE or implemented condition and a zero (0) representing a FALSE or not implemented condition.

5.6.2 Extracted Section from the CANDIDATE DOCUMENT to Compare

9.8.3 Table 72 Events Identification Table

Table 72 Data Description

EVENTS_ID_TBL (Table 72) contains the events that are supported by the End Device. Any of these events may expressed in the History Log Data (Table 74) and the Event Log Data (Table 76).

Global Default Table Property Overrides: Role="CONTROL"

Table 72 Type Definitions

TYPE EVENTS_SUPPORTED_RCD = PACKED RECORD
STD_EVENTS_SUPPORTED : SET(ACT_LOG_TBL.NBR_STD_EVENTS);
MFG_EVENTS_SUPPORTED : SET((ACT_LOG_TBL.NBR_MFG_EVENTS * 8 + 7) / 8);
END;

TABLE 72 EVENTS_ID_TBL = EVENTS_SUPPORTED_RCD;

Table 72 Element Descriptions

Identifier	Value	Definition
EVENTS_SUPPORTED_RCD		
STD_EVENTS_SUPPORTED		This set Element indicates which of the standard events are supported in the Event Log. See Annex B, "History & Event Log Codes", for standard event codes. Event codes are represented by bits 0 through (8 * **ACT_LOG_TBL.NBR_STD_EVENTS** - 1), with a one (1) representing a TRUE or implemented condition and a zero (0) representing a FALSE or not implemented condition.
MFG_EVENTS_SUPPORTED		This set Element indicates which of the manufacturer events are supported in the Event Log. Events are enabled by bits 0 through (8 * **ACT_LOG_TBL.NBR_MFG_EVENTS** - 1), with a one (1) representing a TRUE or implemented condition and a zero (0) representing a FALSE or not implemented condition.

5.6.3 Section from the CANDIDATE TDL/XML Form used to Produce the CANDIDATE DOCUMENT

```
[1]  …
[2]
[3]  <table name="EVENTS_ID_TBL" number="72" type="EVENTS_SUPPORTED_RCD"
[4]        label="Events Identification Table" role="CONTROL">
[5]        <description>
[6]           <b>EVENTS_ID_TBL</b> (Table 72) contains the events that are supported by the End Device. Any of
[7]                 these events may be expressed in the History Log Data (Table 74) and the Event Log Data (Table 76).
[8]        </description>
[9]        <packedRecord name="EVENTS_SUPPORTED_RCD">
[10]          <set name="STD_EVENTS_SUPPORTED" dimension="ACT_LOG_TBL.NBR_STD_EVENTS">
[11]             <description>This set Element indicates which of the standard events are supported in the Event Log. See
[12]                Annex B, &#8220;History &amp; Event Log Codes&#8221;, for standard event codes. Event codes are
[13]                represented by bits 0 through (8 * <b>ACT_LOG_TBL.NBR_STD_EVENTS</b> - 1), with a one (1)
[14]                representing a TRUE or implemented condition and a zero (0) representing a FALSE or not
[15]                implemented condition.
[16]             </description>
[17]          </set>
[18]          <set name="MFG_EVENTS_SUPPORTED" dimension="ACT_LOG_TBL.NBR_MFG_EVENTS * 8">
[19]             <description>This set Element indicates which of the manufacturer events are supported in the Event Log.
[20]                Events are enabled by bits 0 through (8 * <b>ACT_LOG_TBL.NBR_MFG_EVENTS</b> - 1), with a one
[21]                (1) representing a TRUE or implemented condition and a zero (0) representing a FALSE or not
[22]                implemented condition.
[23]             </description>
[24]          </set>
[25]       </packedRecord>
[26] </table>
[27] …
[28]
```

5.6.4 Solution – Issues to be identified and reported by the ACCREDITOR

9.8.3 Table 72 Events Identification Table

Table 72 Data Description

EVENTS_ID_TBL (Table 72) contains the events that are supported by the End Device. Any of these events may expressed in the History Log Data (Table 74) and the Event Log Data (Table 76).

Global Default Table Property Overrides: Role="CONTROL"

Table 72 Type Definitions

TYPE EVENTS_SUPPORTED_RCD = PACKED RECORD
STD_EVENTS_SUPPORTED : SET(ACT_LOG_TBL.NBR_STD_EVENTS);
MFG_EVENTS_SUPPORTED : SET((ACT_LOG_TBL.NBR_MFG_EVENTS * 8 + 7) / 8);
END;

TABLE 72 EVENTS_ID_TBL = EVENTS_SUPPORTED_RCD;

Table 72 Element Descriptions

Identifier	Value	Definition
EVENTS_SUPPORTED_RCD		
STD_EVENTS_SUPPORTED		This set Element indicates which of the standard events are supported in the Event Log. See Annex B, "History & Event Log Codes", for standard event codes. Event codes are represented by bits 0 through (8 * **ACT_LOG_TBL.NBR_STD_EVENTS** - 1), with a one (1) representing a TRUE or implemented condition and a zero (0) representing a FALSE or not implemented condition.
MFG_EVENTS_SUPPORTED		This set Element indicates which of the manufacturer events are supported in the Event Log. Events are enabled by bits 0 through (8 * **ACT_LOG_TBL.NBR_MFG_EVENTS** - 1), with a one (1) representing a TRUE or implemented condition and a zero (0) representing a FALSE or not implemented condition.

5.6.5 Solution – Section from the CANDIDATE TDL/XML Form used to Produce the CANDIDATE DOCUMENT

```
[1]  …
[2]
[3]  <table name="EVENTS_ID_TBL" number="72" type="EVENTS_SUPPORTED_RCD"
[4]       label="Events Identification Table" role="CONTROL">
[5]       <description>
[6]          <b>EVENTS_ID_TBL</b> (Table 72) contains the events that are supported by the End Device. Any of
[7]               these events may be expressed in the History Log Data (Table 74) and the Event Log Data (Table 76).
[8]       </description>
[9]       <packedRecord name="EVENTS_SUPPORTED_RCD">
[10]         <set name="STD_EVENTS_SUPPORTED" dimension="ACT_LOG_TBL.NBR_STD_EVENTS">
[11]            <description>This set Element indicates which of the standard events are supported in the Event Log. See
[12]               Annex B, &amp;#8220;History &amp;amp; Event Log Codes&amp;#8221;, for standard event codes. Event codes are
[13]               represented by bits 0 through (8 * <b>ACT_LOG_TBL.NBR_STD_EVENTS</b> - 1), with a one (1)
[14]               representing a TRUE or implemented condition and a zero (0) representing a FALSE or not
[15]               implemented condition.
[16]            </description>
[17]         </set>
[18]         <set name="MFG_EVENTS_SUPPORTED" dimension="ACT_LOG_TBL.NBR_MFG_EVENTS * 8">
[19]            <description>This set Element indicates which of the manufacturer events are supported in the Event Log.
[20]               Events are enabled by bits 0 through (8 * <b>ACT_LOG_TBL.NBR_MFG_EVENTS</b> - 1), with a one
[21]               (1) representing a TRUE or implemented condition and a zero (0) representing a FALSE or not
[22]               implemented condition.
[23]            </description>
[24]         </set>
[25]      </packedRecord>
[26] </table>
[27] …
[28]
```

The CANDIDATE DOCUMENT exhibits the following errors:

- Line 10 (of TDL/XML Form): The TDL/XMl Form file contains a dimension that is 1/8 of the desired dimension. Found `ACT_LOG_TBL.NBR_STD_EVENTS`, expect `ACT_LOG_TBL.NBR_STD_EVENTS * 8`.

Note: The apparent discrepancy between line 14 of the referenced standard Document Form and line 14 of the DOCUMENT is not an error. In actual fact, it is a correct rendering of the referenced standard, when it applies to the special expression of SETs. SETs are dimensioned in the DOCUMENT in multiple of octets, where each octet contains 8-bits (Boolean flags). On the other hand, `<set>`s are dimensioned in bits (Boolean flags). Therefore when rendering a DOCUMENT from TDL/XML Form, it is necessary to divide the dimension expressed in the TDL/XML Form by eight then round it up to the smallest integer not less computed resulting ratio. Using integer-only algebra resulted in having the expression: **`SET((ACT_LOG_TBL.NBR_MFG_EVENTS * 8 + 7) / 8)`**, in line 14 of the DOCUMENT, that is mathematically equivalent to **`SET(ACT_LOG_TBL.NBR_MFG)`** as shown Line 14 of the referenced standard Document Form.

Note: For the same reason stated above, although the DOCUMENT agrees with the referenced standard Document Form on line 13, it may indicated an error in dimensioning of the `<set>` in the CANDIDATE's TDL/XML Form file. Therefore, when inspecting SETs, then one should check (or at least spot check) that the corresponding dimensioned `<set>` is in of bits and not in octets (i.e. it should multiplied by 8). For more information see "I.2.11.2 <set> Attributes" of Ref. [1] and Ref. [2].

5.7 Exercise 7.

5.7.1 Extracted Section from the Referenced Standard Document Form

9.5.2 Table 41 Actual Security Limiting Table

Table 41 Data Description

ACT_SECURITY_LIMITING_TBL (Table 41) defines the actual number of passwords and security access level entries supported by the End Device.

Global Default Table Property Overrides: Role="ACTUAL"

Table 41 Type Definitions

TABLE 41 ACT_SECURITY_LIMITING_TBL = SECURITY_RCD;

Table 41 Element Descriptions

Identifier	Value	Definition
SECURITY_RCD		**Redefines: DIM_SECURITY_LIMITING_TBL. SECURITY_RCD**.
NBR_PASSWORDS	0	Actual access levels are not established through the Standard password mechanism.
	1..255	Actual number of passwords used by this End Device.
PASSWORD_LEN 0	Reserved.	
	1..20	Actual length of passwords in octets this End Device is supporting.
	21..255	Reserved.
NBR_KEYS	0..255	Actual number of keys this End Device is supporting.
KEY_LEN	0..255	Actual length of keys in octets this End Device is supporting.
NBR_PERM_USED	0..65535	Actual number of user-defined security access entries this End Device supports in addition to those defined through default access permissions.

5.7.2 Extracted Section from the CANDIDATE DOCUMENT to Compare

9.5.2 Table 41 Actual Security Limiting Table

Table 41 Data Description

ACT_SECURITY_LIMITING_TBL (Table 41) defines the actual number of passwords and security access level entries supported by the End Device.

Global Default Table Property Overrides: Role="LIMITING", Accessibility="READONLY"

Table 41 Type Definitions

```
TYPE SECURITY_RCD = PACKED RECORD
        NBR_PASSWORDS       : UINT8;
        PASSWORD_LEN        : UINT8;
        NBR_KEYS            : UINT8;
        KEY_LEN             : UINT8;
        NBR_PERM_USED       : UINT16;
END;

TABLE 41 ACT_SECURITY_LIMITING_TBL = SECURITY_RCD;
```

Table 41 Element Descriptions

Identifier	Value	Definition
SECURITY_RCD		
NBR_PASSWORDS	0	Actual access levels are not established through the Standard password mechanism.
	1..255	Actual number of passwords used by this End Device.
PASSWORD_LEN 0	Reserved.	
	1..20	Actual length of passwords in octets this End Device is supporting.
	21..255	Reserved.
NBR_KEYS	0..255	Actual number of keys this End Device is supporting.
KEY_LEN	0..255	Actual length of keys in octets this End Device is supporting.
NBR_PERM_USED	0..65535	Actual number of user-defined security access entries this End Device supports in addition to those defined through default access permissions.

5.7.3 Section from the CANDIDATE TDL/XML Form used to Produce the CANDIDATE DOCUMENT

```
[1]  …
[2]
[3]  <table name="ACT_SECURITY_LIMITING_TBL" number="41" type="SECURITY_RCD"
[4]           label="Actual Security Limiting Table" role="LIMITING" accessibility="READONLY">
[5]       <description>
[6]          <b>ACT_SECURITY_LIMITING_TBL</b> (Table 41) defines the actual number of passwords and security
[7]               access level entries supported by the End Device.
[8]       </description>
[9]       <packedRecord name="SECURITY_RCD" >
[10]         <element name="NBR_PASSWORDS" type="UINT8">
[11]            <enumerator>
[12]               <enum value="0"
[13]                  text="Actual access levels are not established through the Standard password mechanism." />
[14]               <enum value="1" endValueInclusive="255"
[15]                  text="Actual number of passwords used by this End Device." />
[16]            </enumerator>
[17]         </element>
[18]         <element name="PASSWORD_LEN" type="UINT8" min="0" max="20">
[19]            <enumerator>
[20]               <enum value="0" text="Reserved." exclude="true" />
[21]               <enum value="1" endValueInclusive="20"
[22]                  text="Actual length of passwords in octets this End Device is supporting." />
[23]               <enum value="21" endValueInclusive="255" text="Reserved." exclude="true" />
[24]            </enumerator>
[25]         </element>
[26]         <element name="NBR_KEYS" type="UINT8">
[27]            <description>Actual number of keys this End Device is supporting.</description>
[28]         </element>
[29]         <element name="KEY_LEN" type="UINT8">
[30]            <description>Actual length of keys in octets this End Device is supporting.</description>
[31]         </element>
[32]         <element name="NBR_PERM_USED" type="UINT16">
[33]            <description>Actual number of user-defined security access entries this End Device supports in addition
[34]               to those defined through default access permissions.
[35]            </description>
[36]         </element>
[37]      </packedRecord>
[38] </table>
[39] …
```

5.7.4 Solution – Issues to be identified and reported by the ACCREDITOR

9.5.2 Table 41 Actual Security Limiting Table

Table 41 Data Description

ACT_SECURITY_LIMITING_TBL (Table 41) defines the actual number of passwords and security access level entries supported by the End Device.

Global Default Table Property Overrides: Role="LIMITING", Accessibility="READONLY"

Table 41 Type Definitions

```
TYPE SECURITY_RCD = PACKED RECORD
        NBR_PASSWORDS       : UINT8;
        PASSWORD_LEN        : UINT8;
        NBR_KEYS            : UINT8;
        KEY_LEN             : UINT8;
        NBR_PERM_USED       : UINT16;
END;

TABLE 41 ACT_SECURITY_LIMITING_TBL = SECURITY_RCD;
```

Table 41 Element Descriptions

Identifier	Value	Definition
SECURITY_RCD		
NBR_PASSWORDS	0	Actual access levels are not established through the Standard password mechanism.
	1..255	Actual number of passwords used by this End Device.
PASSWORD_LEN 0	Reserved.	
	1..20	Actual length of passwords in octets this End Device is supporting.
	21..255	Reserved.
NBR_KEYS	0..255	Actual number of keys this End Device is supporting.
KEY_LEN	0..255	Actual length of keys in octets this End Device is supporting.
NBR_PERM_USED	0..65535	Actual number of user-defined security access entries this End Device supports in addition to those defined through default access permissions.

5.7.5 Solution – Section from the CANDIDATE TDL/XML Form used to Produce the CANDIDATE DOCUMENT

```
[1]  …
[2]
[3]  <table name="ACT_SECURITY_LIMITING_TBL" number="41" type="SECURITY_RCD"
[4]           label="Actual Security Limiting Table" role="LIMITING" accessibility="READONLY" >
[5]       <description>
[6]          <b>ACT_SECURITY_LIMITING_TBL</b> (Table 41) defines the actual number of passwords and security
[7]              access level entries supported by the End Device.
[8]       </description>
[9]       <packedRecord name="SECURITY_RCD" >
[10]         <element name="NBR_PASSWORDS" type="UINT8">
[11]            <enumerator>
[12]               <enum value="0"
[13]                  text="Actual access levels are not established through the Standard password mechanism." />
[14]               <enum value="1" endValueInclusive="255"
[15]                  text="Actual number of passwords used by this End Device." />
[16]            </enumerator>
[17]         </element>
[18]         <element name="PASSWORD_LEN" type="UINT8" min="0" max="20">
[19]            <enumerator>
[20]               <enum value="0" text="Reserved." exclude="true" />
[21]               <enum value="1" endValueInclusive="20"
[22]                  text="Actual length of passwords in octets this End Device is supporting." />
[23]               <enum value="21" endValueInclusive="255" text="Reserved." exclude="true" />
[24]            </enumerator>
[25]         </element>
[26]         <element name="NBR_KEYS" type="UINT8">
[27]            <description>Actual number of keys this End Device is supporting.</description>
[28]         </element>
[29]         <element name="KEY_LEN" type="UINT8">
[30]            <description>Actual length of keys in octets this End Device is supporting.</description>
[31]         </element>
[32]         <element name="NBR_PERM_USED" type="UINT16">
[33]            <description>Actual number of user-defined security access entries this End Device supports in addition
[34]               to those defined through default access permissions.
[35]            </description>
[36]         </element>
[37]      </packedRecord>
[38]  </table>
[39]  …
```

The CANDIDATE DOCUMENT exhibits the following errors:

- Line 8: Found attributes `Role="LIMITING", Accessibility="READONLY"`, expected `Role="ACTUAL"`.
- Line 26: Missing reference to redefined SEURITY_RCD from Table 41. Expecting: `Redefines: DIM_SECURITY_LIMITING_TBL. SECURITY_RCD`.

> Note: When data types, such as packed records, bit fields and enumerators are constrained (prototyped) by a referenced data type, as is the case in when redefining) the `SECURITY_RCD` in table 41 the definition of `SECURITY_RCD` of table 40, then the "`Redefines`" attribute SHOULD be present in the description section only. The syntax section of the referencing context SHOULD not duplicate the referenced definition. However the TDL/XML Form file SHOULD contain the full re-definition of the referenced type, so that it MAY document the differences to the application.

Note: A duplication of a type definition in the DOCUMENT MAY be an indication that the type is defined in the TDL/XML Form without proper qualification of it being restricted by a referenced prototype. If left unnoticed it MAY result in an End Device / MDMS / Head-end system failure.

6 References[A1]

1. ANSI C12.19-2008 – ANSI, "*Utility Industry End Device Data Tables*", ANSI C12.19-2008, February 2009.
2. IEEE P1377-2011 – IEEE, Draft "*Standard for Utility Industry Metering Communication Protocol Application Layer (End Device Data Tables)*", IEEE P1377/D9, February 2010.
3. ANSI C12.22-2008 – ANSI, "Protocol Specification For Interfacing to Data Communication Networks", ANSI C12.22, January 2009.
4. IEEE P1703-2010[A2] – IEEE, Draft "Standard for Local Area Network/Wide Area Network (LAN/WAN) Node Communication Protocol to Complement the Utility Industry End Device Data Tables", IEEE P1703/D8, July 23, 2010.
5. NAEDRA POL – NAEDRA, "*Committee Policies and Procedures*", Revision: 0.9, June 24, 2011.
6. Pascal 1990 ISO, "Pascal", ISO/IEC 7185:1990(E).
7. Proposal for testing TDLs
   – NAEDRA, "*Proposal to NAEDRA for a process for testing Registrar TDLs*", A Moise, July 29, 2010
8. RFC 2119 – IETF, "*Key words for use in RFCs to Indicate Requirement Levels*", Bradner, S., BCP 14, RFC 2119, March 1997.
9. XML-2008 – W3C, "*Extensible Markup Language (XML) 1.0 (Fifth Edition)*", W3C Recommendation 26 November 2008.
10. XHTML-2002 – W3C, "*XHTML 1.0 The Extensible HyperText Markup Language (Second Edition)*", W3C Recommendation 26 January 2000, revised 1 August 2002.