



PAP06 UML Meta Model and EDL White Paper

Disclaimer

The ANSI C12.19 Standard, in printed form or in electronic (.pdf) form is the authoritative document. The IEEE Std 1377™ is an identical version of model content of the standard published by IEEE. The ANSI C12.19 Standard defines the naming conventions and normative object model to be used. No design expectation exists for a round-trip conversion between the printed ANSI C12.19 standard and the UML and/or Schema used in this paper. In fact, it is strongly recommended that those interested in exploring the UML and/or Schema first obtain and read the printed most recent release of the ANSI C12.19 and IEEE Std 1377 Standards. The UML and Schema were not produced, nor endorsed, nor approved, by the ANSI C12 Subcommittee 17 Working Group 2 or IEEE SCC31 Working Group P1377 which were instrumental in the development of ANSI C12.19 and IEEE Std 1377, respectively.

Published: August 27, 2012

1 Problem Statement

The goal of the Smart Grid Interoperability Panel (SGIP) Priority Action Plan (PAP) 06 (“PAP06”) is to develop and publish a process for industry stakeholders to be able to translate the American National Standards Institute (ANSI) C12.19 End Device (meter) data model to and from a common form that will allow the semantics of this and End Device models in other standards to be more readily compared and harmonized. A key objective is to facilitate, as much as possible, the lossless transformation to or from other common forms that may be prevalent in each Smart Grid domain [10]. The PAP06 output develops a representation of the ANSI C12.19 data model using a restricted subset of Unified Modeling Language (UML) as well as a tailored subset of C12.19. For brevity, we refer collectively to the ANSI C12.19, Institute of Electrical and Electronics Engineers (IEEE) IEEE Std 1377 and MC12.19¹ standards as ANSI C12.19.²

While there are currently several "meter models" standards in existence, including ANSI C12.19 [1], IEEE Std 1377 [2], Device Language Message Specification (DLMS)/Companion Specification for Energy Metering (COSEM) / International Electrotechnical Commission (IEC) 62056 [5][6], IEC 61968 [3] and IEC

¹ The MC12.19 2003 document is available (it is known as the “Utility Industry User’s Guide”). Contact the PAP06 Technical Champion for details.

² In fact, the ANSI C12.19 model and its corresponding simplified UML model represent the union of the object models defined both in IEEE Std 1377™ / ANSI C12.19 as expanded by IEEE Std 1703™ / ANSI C12.22. At the time of publication of this paper, the most recent standards are IEEE Std 1377™ and IEEE Std 1703™ that were published after the publication of ANSI C12.19-2008 and ANSI C12.22-2008, respectively. For this reason, the UML model, although named here for convenience after the ANSI C12.19 standard, is in actual fact a realization of IEEE Std 1377™ and IEEE Std 1703™ standards.



61850 [4], this work focuses only on ANSI C12.19 [1]/IEEE Std 1377 [2], generically referenced from here as ANSI C12.19.³

The ANSI C12.19 standard organizes the data and operating criteria to be conveyed into and out of those defined End Devices (e.g., a meter or a control point). The highest level groupings of information are called “Decades”. Decades organize information by subject such as register data, timing, or load profile. Their contained objects are called “Tables.” A large number of Tables are defined to allow representation of many types of operational metering and communication needs. Tables exist for standard common data elements that are canonized by the standards; these are known as “Standard Tables.” Tables that are incorporated into the End Device that are not canonized by the standards are referred to as “Manufacturer Tables.”

As the Smart Grid requires interoperation between meters and many other applications and services, the existence of unique forms of data representation pertinent to a single actor is problematic, requiring complex gateways to translate this representation into alternate formats for information sharing.

This work has the potential to substantially reduce the labor costs of integrating large-scale systems that use the End Device model of ANSI C12.19 (meters, sensors and network devices). By reducing or eliminating the amount of human intervention required, utilities can focus on products and services that provide benefit to the organization and to the customer, rather than spending extensive effort on simply achieving connectivity between computer systems. The ability to share the resources represented by metering can greatly enhance the operation of the energy generation and delivery operations as well as opening up new ways to serve customers. The interoperability framework promoted by ANSI C12.19 and documented in this work can enable utility enterprise-level sharing and support a variety of new applications.

By representing the Tables in a common form, translations can be performed from the ANSI C12.19 form to any alternate form. The use of UML as a common form for interpretation may be geometrically smaller in scope than having to translate directly pairwise between ANSI C12.19 and a number of alternate information models (that are not explicitly ANSI C12.19 or Extensible Markup Language, or XML, based) where no single standard is agreed to be the common form. Note, however, that each time a translation is performed between the ANSI C12.19 data model and any other, there is a potential for loss of information or context, due to the limitations that may exist in the correspondence to the vocabulary of the target model. Additionally, UML cannot universally represent all semantic models not previously constrained to its expressive abilities.

³ The reason for the multitude of standard numbers is the existence of a Memorandum of Understanding (MOU) that was signed by The Association of Electrical Equipment and Medical Imaging Manufacturers, known as “NEMA,” (for ANSI SC17), IEEE (for Standards Coordinating Committee, or SCC, 31) and Measurement Canada (for the Measurement Canada Task Force for Electronic Metering Devices). The purpose of the MOU is “*To develop a standard for Protocol Specification for Interfacing to Data Communication Networks, jointly...*”, because “*it is the belief of the organizations that opportunities exist to coordinate with each other in the independent development and publication of the Work which will provide a benefit to the end users.*” [Ref. C12.19 MOU, 2007] For these reasons, “ANSI C12.19” or “C12.19” is used in this white paper to refer collectively to the ANSI C12.19, IEEE Std 1377TM and MC12.19 standards.



2 Overview of the ANSI C12.19 Standard and models

One of the many challenges in authoring a standard on paper is that it may be difficult to express certain concepts in a manner that is within the grasp of the reader. This is compounded by the fact, in the case of the standard undertaken here, that the standard uses both the English language and a style known as pseudo-code⁴. It is the latter that many find difficult to follow as pseudo-code is often a language in-between standard English and a programming language, intended to capture the main points of what one will eventually put into that programming language, without strictly adhering to rules of that language. The ANSI C12.19 Standard, like many protocol standards, uses a notation that expresses the protocol in a manner that allows for strict rules to be applied, but also then requires some effort to comprehend beyond that of plain English language. This follows good practice for a standard as the goal is to make varying interpretations of the printed standard few, if any, and which is certainly not the case for prose English.

The first versions of the ANSI and IEEE Standards, published in 1997 and 1998, respectively, were authoritative only in their printed form. This then led to some difficulties in aligning the varying implementations of that protocol by each developer. The second versions of the ANSI and IEEE Standards, published in 2008 and 2012, respectively, contained a set of rules and framework that use XML [15] for expressing the printed document. The standards also define vocabulary, the End Device Exchange Language (EDL), that uses XML to express End Device data in a manner that is independent from the underlying communication protocols or the binary encoding of the Element data. Finally, the XML Schema language [16] allows for the syntactic description of the EDL data for a specific meter.

Visuals, such as graphs, diagrams, and the like are employed in C12.19 to explain and document the semantic and operational framework. However, it may be a non-trivial exercise to diagram all of the relationships among portions of the Standard such as the Tables, the Elements, Sub-elements and their dependencies. They are sometimes difficult to track without the use of an XML modeling tool (the use of XML and available tools is outside the scope of this paper). The UML [12] is a standardized modeling language that uses well-defined visual relationships to express complex systems. The power of UML is the common, simple method of constructing the diagrams used to express the complex systems which eliminates the need to interpret each authors' particular graphic style.

Version 1.0 of the ANSI C12.19 Standard was published originally in 1997 as ANSI C12.19-1997 and correspondingly in 1998 as IEEE Std 1377-1998™. Both standards are essentially identical, except for the front matter. Version 2.0 of ANSI C12.19 was published in 2009 as ANSI C12.19-2008. An important difference between versions 1.0 and 2.0 of the standard is the introduction of the XML-based object-oriented Table Definition Language (TDL) that is an abstract End Device class modeling framework (meta-data model). Its data realization companion is the enterprise XML-based EDL, which is used to model End Device Table classes and convey instances of ANSI C12.19 End Device Table objects in a manner that is communication protocol agnostic. The following summarizes these XML technology-based formats:

- TDL An XML description of the table model

⁴ One element of pseudo-code expression is that it "...may vary widely in style from a near-exact imitation of a real programming language at one extreme, to a description approaching formatted prose at the other." [Wikipedia]



- EDL An XML content of meter data
- EDL Schema An XML Schema description of an EDL file which is based on a TDL

The XML-based TDL meta-models are machine readable and can be used by Head-end systems, meter data management systems (MDMS) and other so-called enterprise applications to interpret the ANSI C12.19 End Device data elements when translating Meter Data Management System (MDMS) Common Information Model (CIM)⁵-based messages to Head-end/ANSI C12.19 Tables, and by the Head-end systems to manage the real-time communication with the End Devices (e.g. meters). The XML-based EDL classes (expressed as EDL schema files) and objects and the elements (expressed as XML file) hold machine readable forms that can be used by Head-end systems and MDMS enterprise applications to convey ANSI C12.19 End Device element data.

Figure 1 illustrates a fragment of the pseudo-code representation of the C12.19 data model. Figure 2 shows how that same fragment would be represented in the TDL. Finally, Figure 4 shows how that fragment might be represented in the EDL Schema.

```

TYPE ED_MODE_BFLD = BIT FIELD OF UINT8
    METERING_FLAG           : BOOL(0);
    TEST_MODE_FLAG         : BOOL(1);
    METER_SHOP_MODE_FLAG   : BOOL(2);
    FILLER                  : FILL(3..7);
END;

```

Figure 1: Example ANSI C12.19 Syntax

```

<bitField name="ED_MODE_BFLD" type="UINT8">
  <subElement name="METERING_FLAG" type="BOOL" startBitInclusive="0">
  </subElement>
  <subElement name="TEST_MODE_FLAG" type="BOOL" startBitInclusive="1">
  </subElement>
  <subElement name="METER_SHOP_MODE_FLAG" type="BOOL" startBitInclusive="2">
  </subElement>
  <subElement name="FILLER" type="FILL" startBitInclusive="3" endBitInclusive="7">
  </subElement>
</bitField>

```

Figure 2: Example XML code derived from ANSI C12.19 rules (TDL)

⁵ The Common Information Model referred to in the white paper is that of the IEC Technical Committee 57, captured in the IEC 61970 and IEC 61968 suites of standards. <http://www.iec.ch/>



```

<xsd:complexType name="ED_MODE_STATUS_TBL.ED_MODE_BFLD">
  <xsd:sequence>
    <xsd:element name="METERING_FLAG" type="ED_MODE_STATUS_TBL.ED_MODE_BFLD.METERING_FLAG" minOccurs="0"
      maxOccurs="1" />
    <xsd:element name="TEST_MODE_FLAG" type="ED_MODE_STATUS_TBL.ED_MODE_BFLD.TEST_MODE_FLAG"
      minOccurs="0" maxOccurs="1" />
    <xsd:element name="METER_SHOP_MODE_FLAG"
      type="ED_MODE_STATUS_TBL.ED_MODE_BFLD.METER_SHOP_MODE_FLAG" minOccurs="0" maxOccurs="1" />
    <xsd:element name="FACTORY_FLAG" type="ED_MODE_STATUS_TBL.ED_MODE_BFLD.FACTORY_FLAG" minOccurs="0"
      maxOccurs="1" />
    <xsd:element name="FILLER" type="ED_MODE_STATUS_TBL.ED_MODE_BFLD.FILLER" minOccurs="0" maxOccurs="1" />
  </xsd:sequence>
  <xsd:attributeGroup ref="ElementIndexAttr" />
</xsd:complexType>

```

Figure 3: Example XML Code Derived From ANSI C12.19 Rules (EDL Schema)

```

<xsd:complexType name="EdModeStatusTbl.EdModeBfld">
  <xsd:sequence>
    <xsd:element name="MeteringFlag" type="EdModeStatusTbl.EdModeBfld.MeteringFlag"
      minOccurs="0" maxOccurs="1"/>
    <xsd:element name="TestModeFlag" type="EdModeStatusTbl.EdModeBfld.TestModeFlag"
      minOccurs="0" maxOccurs="1"/>
    <xsd:element name="MeterShopModeFlag" type="EdModeStatusTbl.EdModeBfld.MeterShopModeFlag"
      minOccurs="0" maxOccurs="1"/>
    <xsd:element name="FactoryFlag" type="EdModeStatusTbl.EdModeBfld.FactoryFlag"
      minOccurs="0" maxOccurs="1"/>
    <xsd:element name="Filler" type="EdModeStatusTbl.EdModeBfld.Filler" minOccurs="0" maxOccurs="1"/>
  </xsd:sequence>
  <xsd:attributeGroup ref="Elementindexattr"/>
</xsd:complexType>

```

Figure 4: Example XML CamelCase Converted from ANSI C12.19 formal EDL Schema

2.1 Standards and Releases

A new maintenance release of the ANSI C12.19 and IEEE Std 1377™ standards is under work by both organizations. The new version of IEEE Std 1377-2012 is backward compatible with ANSI C12.19-2008, which in turn is backward compatible with ANSI C12.19-1997.

The current version of ANSI C12.19 is 2.0 and IEEE Std 1377-2012 will be considered version 2.1. Version 2.1 corrects errors that were discovered since the publication and deployment of version 2.0. Version 2.1 also targets End Device interoperability by including a tie (new device class object identifier) to a more constrained data model (meter profile) that is based on a publication of the Association of Edison Illuminating Companies (AEIC) entitled “Smart Grid/AEIC AMI⁶ Interoperability Standard Guidelines for ANSI C12.19 / IEEE Std 1377 / MC12.19 End Device Communications and Supporting Enterprise Devices, Networks and Related Accessories.” This reference model is more commonly known as the “AEIC Guidelines V2.0” [11]. The AEIC Guidelines V2.0 is the work product of AEIC Advanced Metering Technology Issues Team (AMTIT) and Smart Grid Interoperability Panel (SGIP) PAP05. The data model

⁶ Advanced Metering Infrastructure



that is presented in this paper is based on IEEE Std 1377-2012, which is backward compatible with ANSI C12.19-2008, which in turn is backward compatible with ANSI C12.19-1997 (and, correspondingly, IEEE Std 1377-1998).

2.2 Tables, Decades and other Elements

As introduced earlier, the ANSI C12.19 standard defines a data model that is used to exchange information among electric, water, and gas meters (one type of an “End Device”), or ANSI C12.22 Hosts, Relays and Gateways (another type of an “End Device”) as well as support the semantic needs of enterprise Head-end systems (yet another type of an “End Device”). The data model consists of Tables that are grouped (packaged) into Decades. Each Decade represents a common End Device’s function-set and associated features. The data Tables are extensible, with considerable space reserved for future innovation both through the publication of revisions to the standard, and by the users (e.g. meter vendors) through the release of new third party extensions (generalizations) to the ANSI C12.19 published data models (standard End Device classes).

From a data model stand-point, ANSI C12.19 supports a maximum of 2039 Standard defined Tables, and 2039 Manufacturer Tables. The Table number (0 – 2039 Standard tables and 0 – 2039 Manufacturer tables) is used as a selector in a communication protocol, such as ANSI C12.22. The Tables associated with this selector can be modeled using the ANSI C12.19 modeling framework and they can be listed in Standard Table zero, or “00,” that is the End Device “General Configuration Table” – the End Device’s table of contents. Therefore, Formal Tables can be expressed in the EDL file. The standard also provides a capability to aggregate data elements from multiple Tables in real-time through the use of “user-defined tables” and “extended user-defined tables” controls and selectors, which behave like macros that provide custom views.

The standard envisions all exchanges with “end devices” to be via simple SET and GET methods. Many modern protocols use this “reduced instruction set” approach to modeling because it results in a flexible and easily extensible (just add definitions of new things to SET and GET) architecture. These concepts are commonly referred to as Create/Read/Update/Delete (CRUD⁷) or REpresentational State Transfer (REST⁸). More complex formulations add new coding requirements to the protocol exchange when an extension is made.

ANSI C12.19, therefore, defines three abstract⁹ communication services, or methods, for getting and setting Table data element values. Although these methods and the encoding are beyond the scope of the present modeling activity, it is discussed here briefly for context. These are the Table Read (full or partial), Table Write (full or partial) and Procedure Initiate methods (that is a variation on the Table Write method). These methods of communication (Table Read and Table Write) utilize binary encoded parameters and payloads when using the protocols for the conveyance of this data via ANSI C12.18, ANSI C12.21, and ANSI C12.22.

⁷ See http://en.wikipedia.org/wiki/Create,_read,_update_and_delete

⁸ See http://en.wikipedia.org/wiki/Representational_state_transfer

⁹ They are abstract to imply a template for transport/syntax implementing standards such as C12.18, C12.19, and C12.22 that define discrete protocols for transferring meter data.



When communicating with an End Device using a binary communication protocol (e.g., ANSI C12.22) for the purpose of setting End Device Table element values, the action can be expected to be immediate, which may not always be a desirable result. For this reason ANSI C12.19 supports a “Pending Table” communication attribute together with a pending table activation trigger record, which can be used to defer the activation (use) of the values transmitted to an End Device (e.g., a meter) until pre-set or other triggering condition is satisfied. This feature facilitates End Device group programming, End Device firmware upgrades with activation and roll-back support. Pending Tables also facilitate event-driven or synchronized actionable requests issued by enterprise systems (such as Head-end systems) that communicate with a multitude of C12.19 devices (e.g., via multicast) in an AMI network of a Smart Grid.

In order to provide a clear means of exchange of instance models of actual End Devices, C12.19 defines the notion of an End Device Class represented by a universally unique object identifier (OID). Each unique OID represents a unique selection of model components, decades and tables that can be defined formally in the TDL and EDL. These OIDs can be held in a registry along with TDL and EDL descriptions.



3 Introduction to UML

The UML is intended to be a means to establish a common modeling basis for a variety of readers. The selection of a UML-based model has the following benefits:

- It is an information modeling-knowledge repository. For example, contrast a drawing program like Microsoft Visio™ with a dedicated schematic capture tool for making a model of a circuit. Because the schematic tool understands that there are components with connections which are inputs and outputs, for example, this information can be used to manage consistency of the use of the parts library – inputs must only be connected to outputs, for example. A graphic-only tool provides for rectangles and circles and such and the knowledge of other relevant properties are excluded.

Figure 5 contains an example of a UML diagram relating certain elements of a unified model in what is called a Use Case diagram.

- UML is an international standard specifically designed to allow for the description of complex systems. Note that there are many presentations with stunning commercial graphics attempting to depict the author's thinking and analysis. Each time the audience must accommodate themselves to the style of the artist and understand not just what is rendered but the style with which it was rendered. With UML, however, there is defined a common and simple way to make such drawings. Once the particular drawing methods and conventions have been learned, the meaning does not vary with each viewing of a model. Figure 6 illustrates this concept.
- Once information is captured in a machine readable way, it can readily be rendered and exported to other tools for various uses. The eXtensible Markup Language Metadata Interchange (XMI) form of a UML model is an open formal standardized format. It can be interrogated within a UML editor designed for this purpose. In addition, various Web tools that are designed for interaction with XML files based on XML Schemas can be used to provide custom and powerful interfaces to the "XMI Data Base" (see Figure 6).
- Because UML is a modeling language, its components are re-useable in describing other and future concepts built on a library of well understood building blocks. This is similar to how standard, well-known shapes can be used to construct a facsimile of virtually any object the mind can envision. [13]

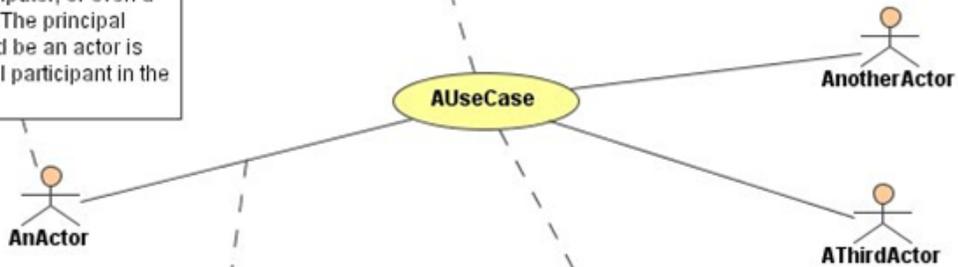
Note: One caveat to all of this is that there remain some challenges in working with XML schema and the lossless and identical import and export functions with various UML tools. It is still possible for two different UML tools to render different results when importing the same schema. It is critical that those dealing with standards, schema and UML models pay particular attention to the exact revision numbers of all of the tools being used to gain confidence that the end results of two different users are identical.



A Use Case provides an overview of the functional requirements for the application of technology. Use Cases expose functional requirements, which should be distinguished from design requirements. Whereas functional requirements identify "what the customer wants", design requirements define those technical components that can combine to achieve the functional requirements.

Use Cases can be simple or elaborate. A simple Use Case can be a single statement - "We need to represent the name, model, and serial number of a device to facilitate commerce". An elaborate use case can be volumes of documentation. Intelligrid strives to strike a balance between complexity and ease of presentation of information.

A stick figure that represents one of the participants in the Use Case. An actor may be a user, a computer, or even a program or process. The principal choice of what should be an actor is identifying an external participant in the Use Case.



A line connects or "associates" an Actor with a Use Case. All the Actors participating in a specific Use Case will have a line connecting it with the Use Case symbol.

A horizontal oval symbol that represents a single Use Case

Figure 5: A Use Case Diagram illustrates who is Involved in Function [15]¹⁰

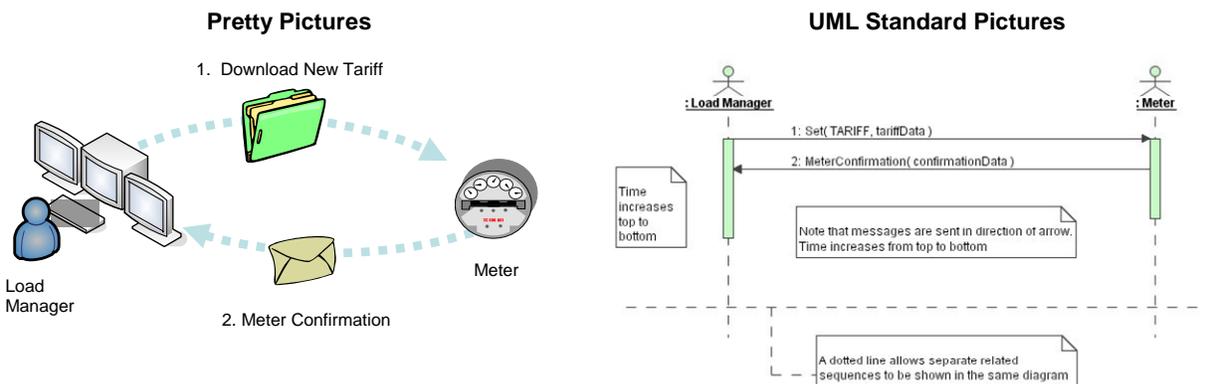


Figure 6: Illustration of a Simple Drawing (left) and a UML Sequence Diagram (right) [15]

¹⁰ The term "Intelligrid" in the example Use Case text refers to the work found at <http://intelligrid.epri.com/>

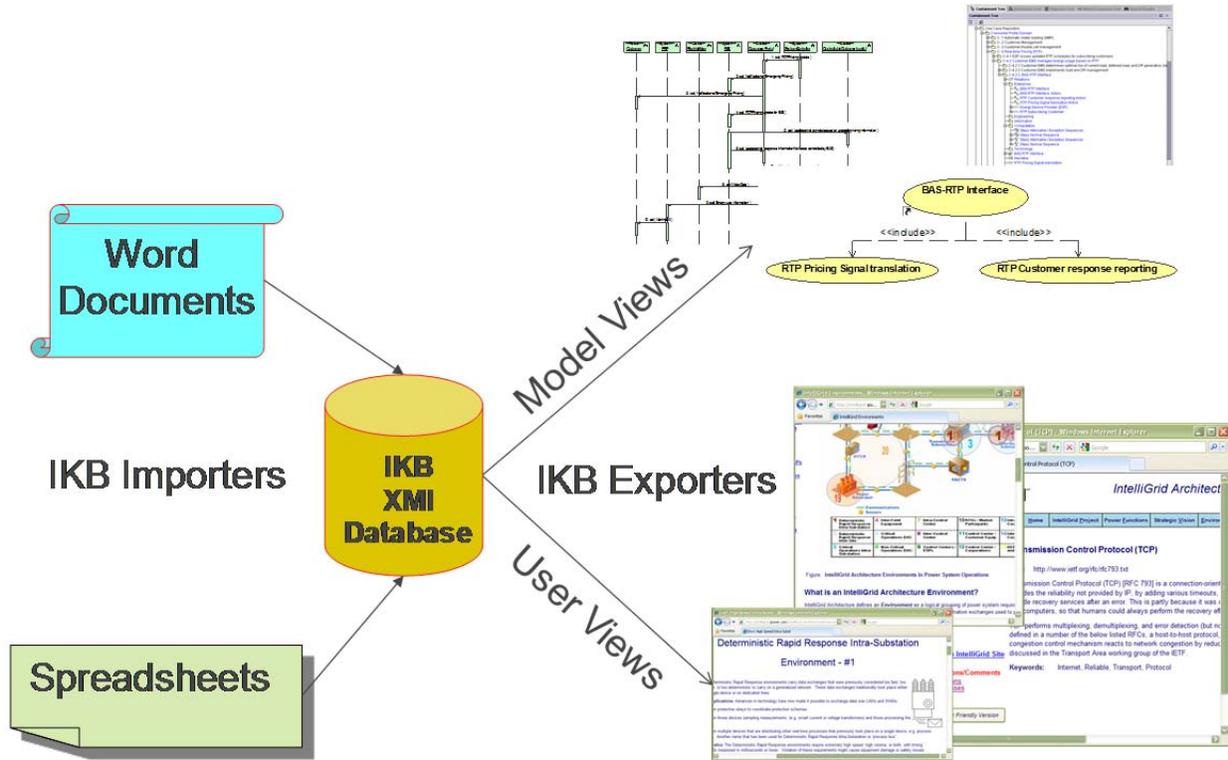


Figure 7: Example of Managing a Database Within a Modeling Tool [15]

Note: IKB is “interoperability knowledge base”



4 The ANSI C12.19 Tables Meta Model

This paper describes the resulting UML-based model that was constructed from the C12.19 standard. The model is presented in two parts – The Meta Model¹¹ and the Data Model. This section describes the Meta Model. In essence, the Meta Model describes in UML terms the basic data types supported by C12.19 Devices and the relational structure of information that may be represented in Tables and Procedure attributes.

In this model, Tables are classes that contain information elements. Only information that is contained inside a Table class can be referenced by the AMI communication system. Tables are associated with End Device functional attributes. For example, one Table may express the attributes of a clock; another Table may express energy measures; and another may hold device identification information. The Table Elements can be communicated individually or collectively aggregated, as needed. Tables that serve similar or related functions are associated with functionality groups; these are called Decades. Traditionally each Decade can be associated with up to 20 Tables (10-Standard and 10-Manufacturer Tables). Although the number of Tables per Decade seems to be limited to 20 Tables, effective in Version 2.0 (or later) the C12.19 Meta Model provides semantics for extending the Decades beyond this artificial limit.

Tables and their elements, on their own, serve simply as place holders for dimensioning, information, control, and status and sensor data content. A special purpose Table class (Table 07, Procedure Initiate) is also defined by the standard Meta Model. The Procedure Initiate Table's Elements act as triggers and accept parameters that cause various standard defined End Device procedures or vendor supplied Procedures to be executed. Procedures have a name and a number, and are manifested by overloading Table 07 (where the control attribute is the procedure number). For example, Procedure 06, Change End Device Mode, is used to manage the operational state of a meter, and Procedure 09, Remote Reset, is used to perform a commodity peak demand measurement reset.

The Meta Model also furnishes meta-Table data, meta-Procedure data and meta-Element data. These provide additional attributes for Tables, Procedures and their Elements that may be needed by an implementation to interact with the C12.19 Device in a manner that is interoperable and reliable. Examples of meta-Table data may be the Table role, group association, accessibility, volatility, etc.

ANSI C12.19 Tables are class instances of Packed Records¹². Packed Records can contain Elements that refer to other Packed Records, Bit Field records and any elementary data types. Elementary data types (such as INT8, FLOAT64 or STRING) are known as "Final Elements". Within an ANSI C12.19 device class context (the term used to denote a specific profile of the standard), the Decades, Tables and Procedures act as containers for Packed Records, Bit Field records and enumerators. Therefore, Decades, Tables and Procedures are both class and containers for inner class definitions of Packed Records, Bit Fields and

¹¹ Wikipedia: "Metamodeling, or meta-modeling in software engineering and systems engineering among other disciplines, is the analysis, construction and development of the frames, rules, constraints, models and theories applicable and useful for modeling a predefined class of problems."

¹² "Packed Records" is a term originally introduced by the pseudo-Pascal language that is used to publish human-readable "Document Form" of the ANSI C12.19 standard.



Enumerators. However, only Tables are intended to be instantiated into objects, and only Tables (or elements of Tables) may carry information or be acted upon. For reference, Figure 8 shows the C12.19 Meta Model in UML.

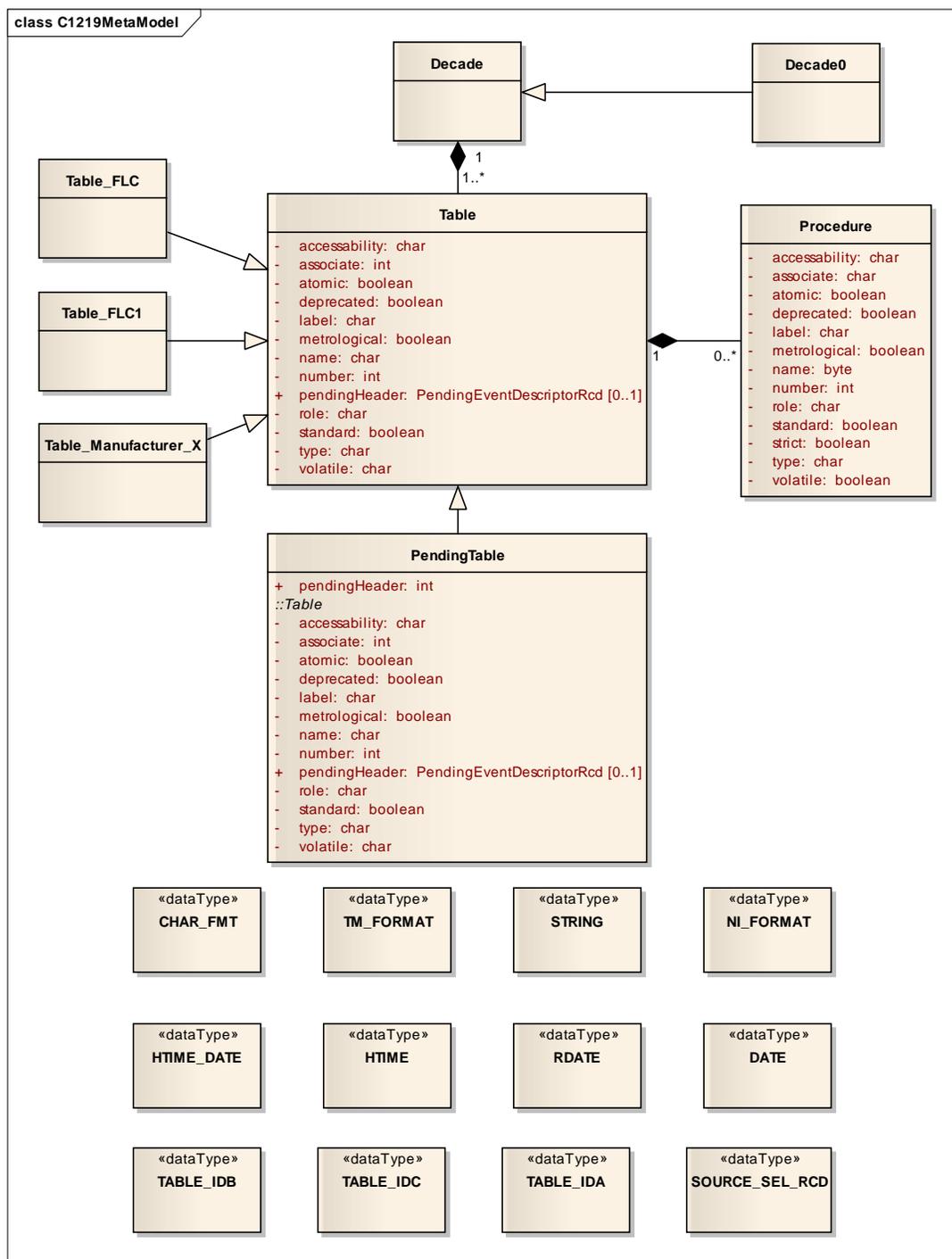


Figure 8: Meta Model Overview



4.1 C12.19 Model Structure

C12.19 was designed to support highly configurable End Devices. Given that actual End Devices have limits in hardware and software, the standard was designed to facilitate the advertising of the limits of configurability of various features.

4.1.1 Decades

Decades group Tables (see Table 1) into functional groupings. Each Decade defines a function in terms of configuration, control and status.¹³

Number	Decade Name	Decade Description
00	GEN_CONFIG_DEC	General Configuration Tables – contains device configuration and identification tables.
01	DATA_SOURCE_DEC	Data Source Tables – contains sensor control, unit of measures and scalars
02	REGISTER_DEC	Register Tables – contains commodity usage readings including simple and complex (e.g. TOU) registers
03	LOCAL_DISPLAY_DEC	Local Display Tables – contains end device display(s) controls
04	SECURITY_DEC	Security Tables – contains users passwords, access roles, and network security keys
05	TIME_OF_USE_DEC	Time and Time-of-Use Tables – contains clocks, schedules and calendars
06	LOAD_PROFILE_DEC	Load Profile Tables – contains multi-channel interval recorded of commodity usage
07	HISTORY_EVENT_DEC	History Log & Event Log Tables – contains information logs of end device activity and managed end device change history
08	USER_DEFINED_TABLES_DEC	User-defined Tables – contains tables that aggregate simple views into other tables to minimize communication burden
09	TELEPHONE_CONTROL_DEC	Telephone Control Tables – contains information, control and status tables for the management of telephony-based (MODEM) in support of ANSI C12.21 / IEEE Std 1702
10	UNASSIGNED	
11	LOAD_CONTROL_DEC	Load Control and Pricing Tables – contains control and status information in support of premise direct load control and pre-payment metering
12	NETWORK_CONTROL_DEC	Node Network Control Tables – contains interface control, status, filtering and information tables of C12.19 Nodes that are deployed on ANSI C12.22 / IEEE Std 1703 networks (the details of these tables can be found in ANSI C12.22 / IEEE Std 1703)
13	NETWORK_RELAY_CONTROL_DEC	Network Relay Control Tables – contains control, status, registration and route information needed by C12.19 Nodes that implement ANSI C12.22 / IEEE Std 1703 Relays and/or Master Relays (the details of these tables can be found in ANSI C12.22 / IEEE Std 1703)

¹³ Note the information in this table is directly from the Standards using the printed document syntax.



Number	Decade Name	Decade Description
14	EXT_UDT_DEC	Extended User-defined Tables – contains tables that aggregate more complex views into meta-tables (tables that have a specialized handle/selector) to minimize communication burden
15	QUALITY_SERVICE_DEC	Quality-of-service Tables – In the case of electrical services, these Tables are the placeholders of Power Quality monitoring controls and data values. In the context of water and gas industries, these Tables are the placeholders of for reports on water and gas commodity quality (such as purity) and sustained delivery of the commodities.
16	ONEWAY_DEVICES_DEC	One-way Devices Tables – contains control, status, and values to manage non-AMI one-way-device communication (e.g. drive-by meter-data collectors)

Table 1: C12.19 and C12.22 Decade Descriptions

Decade zero (0), GEN_CONFIG_DEC, is a special class because it contains special purpose Tables that provide for the End Device “table of contents”, End Device identification classes, and End Device procedure invocation.

4.1.2 Tables

4.1.2.1 Function Limiting Control Tables

The first Table of each Decade (except for Decade 0) is called the Function Limiting Control (FLC) Table. It defines a super class (and interface) that contains elements (attributes) that describe the device absolute upper limits for dimensions of collections and restrictions on Final Element values that are realized by sub-classes of the FLC tables of a given Decade (i.e., the maximum values permitted for instance variables of an End Device).

Sub-classes of FLC Tables for each Decade are used to express the actual dimensions and control attributes that are instantiated in other Table classes that are associated with or packaged within a Decade. The sub-classes of the FLC Tables are referred to generically as the Actual Limits Tables and they contain control elements that are actually used for run-time configuration of an End Device (e.g. a meter, a relay or a Head-end system). Decades and Tables (and Procedures) have names and are numbered. Table numbers span the range of 00 through 2039, and therefore their container Decade numbers are (typically) in the range of 00 through 203 (the Meta Model facilitates the arbitrary association of a Table with a Decade). In this case, the table number may be outside the range associated with a decade. Similarly, Procedure numbers span the range of 00 through 2039.

FLC Table classes are assigned the numeric value of the Decade number that contains them times ten (10). For example, the “Data Source Dimension Limits Table” is defined inside the class DATA_SOURCE_DEC, “Data Source Tables”, which has an assigned ordinal index value of one (1). Therefore, DIM_SOURCES_LIM_TBL, the class that is associated with the FLC Table, is assigned the ordinal index ten (10) – that is “Table 10.”

4.1.2.2 Actual Data Sources Limiting Control Tables

Actual Data Sources Limiting Control Tables contain the current configured state of the Decade. Actual Data Sources Limiting Control Tables are assigned the ordinal index of its FLC super class plus one (1).



For example, given that DIM_SOURCES_LIM_TBL has the ordinal index value of ten (10), then its subclass FLC+1 Table will be ACT_SOURCES_LIM_TBL, “Actual Data Sources Limiting Table”, which has the ordinal index value of eleven (11). All other Tables instances that are associated with that same Decade package will be assigned ordinal indices in the range of Decade<number> x 10 + 2 through Decade<number> x 10 + 9, when defined within the Decade’s package. They may be assigned any available Table number when “associated” with a Decade (a generalization feature of the C12.19 modeling framework).

4.1.2.3 Procedures Invocation Tables

The ANSI C12.19 Standard enterprise Exchange Data Language (EDL) completely defines an XML/Schema that consists of classes (packed records, bit fields, enumerators and Tables). EDL also defines the elements of instances of Tables objects. These are referred to as data sets. The ANSI C12.19 Procedure classes (defined in the TDL meta-model) are absent by name in the EDL because they are referenced through structure Elements of Table 07, PROC_INITIATE_TBL and Table 08, PROC_RESPONSE_TBL.

4.1.2.4 Pending Tables

Each of these concepts is also present in the UML object model. One should consider how ANSI C12.19 EDL defines these concepts before we continue the discussion on mapping of ANSI C12.19 to or from UML/IEC CIM. EDL realization of the Table classes is through the use of World Wide Web Consortium (W3C)/XML data-elements primitives. For example, the kilowatt-hour (kWh) energy usage registers

“CURRENT_REG_DATA_TBL.TOT_DATA_BLOCK.SUMMATIONS[kWh]”

is expressed using the XML “xsd:double” native type notation. This type is a property of the standard, which is independent of the choice of binary encoding used and communicated by the End Device for the type NI_FMAT1 [Ref. ANSI C12.19-2008, Annex I.4 EDL XML Form Encoding of Final Element Values].

4.1.2.5 Manufacturer Tables

By definition, a Manufacturer Table is a Table whose structure is defined by an individual End Device manufacturer. They should be used to introduce innovations or to provide customer-requested data structures that are not defined by the Standard. It is noted that the Standard also provides for the latter to be accomplished in a limited manner through user-defined Tables (Decade 8) and extended user-defined Tables (Decade 14). Manufacturer Tables also provide a path for new functions to evolve into Standard Tables.

4.1.3 Elements of Tables

ANSI C12.19 classes consist of fields (Elements), methods (Table Read, Table Write and Procedure Invoke), and constructors (Table definitions, Type definitions). Fields (Elements) are communication entities (variables) that have types and ordinal positions bound to them. The types of fields are either primitive such as INT16, FLOAT64, or STRING, or they can be reference collections of types such as ARRAYS or SETs, or references to other classes, such as packed records and bit fields. Methods consist of Table Read, Table Write and Procedures Invocation (through Table Write to PROC_INITIATE_TBL), and each takes parameters and communicates instance Table or Procedure data Elements in the order governed by the ordinal index that is assigned to each Table Element.



ANSI C12.19 TDL meta-classes (packed records, bit fields) and fields (Elements) and Table access methods, and Table instance constructors have visibility (e.g. accessibility) and behavioral modifiers as well (e.g. atomic, volatile, metrological, *etc.*) that govern only the manner an End Device accessory (reader) needs to engage the End Device. ANSI C12.19 supports single inheritance between classes. The members of a class consist of the locally declared members (Elements) for the class as well as the inherited members. Unlike UML, ANSI C12.19 class inheritance capability provides for both unidirectional inheritance (redefine and override, where a subclass is constrained by an interface) and bi-directional (replace, where the superclass is actually mapped to the definition of the sub-class). Class members can also be multiply defined as long as the instance of the class (the Table) resolves all ambiguities.

Although ANSI C12.19 classes can be grouped in packages called Decades, Procedures and Tables, this grouping is meaningful only in the context of the enterprise XML/TDL and XML/EDL processing system. When translated into binary communication access methods (e.g. ANSI C12.22 EPSEM Table Read and Table Write requests), the only information that is exchanged “over the air” is the Table number, Table type, First Table Element ordinal index (or position) and the number of Table Elements requested. This information corresponds to the EDL objects that were instantiated from the EDL schema, where the EDL schema is a derivative artifact that is generated from the TDL meta-model.

Within the enterprise system (e.g. Head-end) the symbolic names of classes, enumerators and contained element are used instead. The qualified name of a class consists of the name of the End Device top-level namespace container class (that may be generically referred to as TDL or, alternatively, it may be aliased to STD, when the container is an ANSI C12.19 Standard container) that is followed by hierarchy of Decade, Table or Procedure class names and, optionally, the contained Element name. Use of qualified names is required to distinguish between two classes with the same names in different packages.

Decade names, Table names, and Procedure names are polymorphic; therefore, they are implicitly cast to their corresponding Decade<number>, Table<number>, or Procedure<number>, respectively. That is, the introduction of a Decade class, Table class or Procedure class is also a C12.19 framework directive to associate the container’s number with an enumerator having the name of the container being defined.

When a class is instantiated, the Table instance variables (Elements) are all created in the instance for each of the non-vanishing fields in the class, within the Table. ANSI C12.19 “vanishing Elements” include arrays of zero length or class’s run-time conditional Elements that are conditionally excluded. The instance variables convey values that comprise the state of an object within an End Device. One may invoke instance methods such as read or write on these objects (Tables) if the Table instance exists and the visibility modifier of the Element enables one to do so, in the given particular scope or security role. The invocation of the Table Read or Table Write methods can only be realized using communication (i.e., via a communicating protocol such as ANSI C12.22). Once the data is communicated, the instance Tables payloads and their Elements are packaged using the ANSI C12.19 Exchange Data Language (EDL) that principally maps the ANSI C12.19 instance binary variables payloads communicated as XML/EDL rendered file equivalent for use in enterprise exchange and configuration management (e.g., by the Head-end system, MDMS, billing system or Device programming tools). The ANSI C12.19 Standard includes two special instance methods that are variations on the read and write methods. These are the Procedure Initiate and Procedure Response methods and may be used to realize complex functions or initiate special End Device activities.



Unlike UML models that are designed primarily to be implemented in programming languages, the ANSI C12.19 Data model defines an extensive meta-model framework for communication, generic expression of data and meta-data for enterprise information exchange. The ANSI C12.19 inheritance rules, uniqueness rules, package definition rules (e.g. the name of a Table, such as GEN_CONFIG_TBL, is both the name of a class and the value of the Table's number).

Note: As such, UML may not be suitable or capable of an exact representation (or modeling) of the C12.19 end device class meta-model and attributes, especially if the UML model is additionally constrained by IEC 61970 CIM rules. For this reason, it is stipulated that under the above assumption the present CIM/UML modeling framework is lacking the capability needed to provide a lossless direct mapping to the ANSI C12.19 data model. On the other hand, the ANSI C12.19 modeling framework can easily express CIM using the TDL/EDL framework. Therefore, it is possible to create define mapping relationships, instead of identity mapping, between IEC CIM and ANSI C12.19 EDL.

4.2 The C12.19 Meta Model in UML

4.2.1 Limitations of UML Representation of the C12.19 Meta Model

The original ANSI C12.19-1997 and IEEE Std 1377-1998 meta-models were created to express the on-air payload encapsulation and encoding for communication needs. The model was expressed in pseudo Pascal notation aiming to link Elements that might exist within Tables to data transmission requirements. The C12.19 Meta Model expresses “possibilities” and “choices”, whereas the transmitted data are the actual End Device selections from the realized objects. These constructs were developed in the 90's where use of “structured modeling” was the norm. UML, XML and Schema did not yet exist at that time in the collective consciousness of the authors of the standards¹⁴. Priority was given to efficient transportation as an overriding concern over alternatives such as “object-oriented modeling.”

The post-2008 publications of these standards (version 2.0) introduced contemporary modeling concepts, processing instructions and the facility to express the C12.19 meta-model using XML while preserving the core modeling and naming assumptions that existed in these standards since their inception.

The core requirements of the ANSI C12.19 model calls for unused Tables to exist in the End-device. Similarly unused conditional Table Elements are not to be transported (nor selected), zero-length arrays or sized Elements (such as STRINGS and BINARYs) are not to be transported (nor selectable), Elements may be polymorphic (i.e., a single Element may be bound to a number of different data types subject to a controlling expression that is resolved at run-time). The polymorphism capability used by ANSI C12.19 permits multiple Elements to have a same name and to co-exist at the same level of the Table model hierarchy when they are uniquely represented in any given defined configuration. These multiple instances get reduced (at run-time) to a single and uniquely identifiable representation of the model making an End Device profile (i.e. meter configuration) for communication (e.g. meter readings). The Element names expressed in the C12.19 model must be formed using capital letters and underscore, where classes and complex types must be suffixed with reserved key words (e.g. GEN_CONFIG_TBL is

¹⁴ It was as early as 1985 that MCL (Metering Control Language) was presented to ANSI as a first draft of ANSI C12.19. The MCL, being method and object oriented, was rejected as it was seen to burden the meters' microprocessors far beyond the technology of that time.



the name of a Table, TABLE_IDA_BFLD is the name of a bit field, CHAR_FORMAT_ENUM is the name of an enumerator, and U_SEC_FRACTION is the name of an Element).

Note: Some of the ANSI C12.19 model expression rules cannot be realized or expressed in the subset of UML that is used in this paper. This is not a general limitation of UML, where C12.19 can be modeled; instead it is a limitation of the UML dialect subset that is used here and by IEC CIM.

Additional challenges result from the uniqueness constraints that were imposed by recent versions of the W3C XML schema. Such constraints are only imposed at run-time on an ANSI C12.19 End Device instance and not imposed on the meta-model for the same End Device. Finally, in order to overcome some of the limitations that are being imposed by some UML graphical and class rendering tools (such as Enterprise Architect) that are used to import the C12.19 EDL Schema, we present in this paper a simplified (reduced) C12.19 EDL Schema (see Section 5, From Standard Syntax to UML, for more details). One of the more obvious transformations is the use of CamelCase notation (a naming convention where multiple words are joined without spaces, and each word is capitalized within the compound) to cast C12.19 Element names to the corresponding UML names. For example, the published C12.19 meta-model Element U_SEC_FRACTION was renamed in the EDL discussed in this paper to USecFraction.¹⁵

4.2.2 Addressing the limitations of UML and W3C schema in the C12.19 Meta Model

Table 2 contains a list of identified issues, whether they are addressed and comments related to their resolution or lack thereof.

¹⁵The C12.19 to CamelCase transformation is reversible because ANSI C12.19 Element names are represented by the following regular expression: *identifier = capLetter(capLetter|digit)* ('_(capLetter|digit)*)**. The construction of a C12.19 identifier does not allow for the underscore character (_) to be placed at the beginning or at the end of a phrase nor repeat without at least one interleaving capital letter (A-Z) or at least one interleaving digit (0-9).



Issue	Addressed?	Comment
Need to solve how to represent the polymorphism in C12.19 case statements where alternatives have the same names	No. See comment.	This issue should be referred to standards working groups for resolution.
Does the UML capture the C12.19 semantics?	Partial. See comment.	This UML captures the Standard's class hierarchy only.
Does the <> capture the Final Element or a derived element?	The former. See comment.	The inner-most element in the EDL schema and the UML equivalent are Final Elements as defined in C12.19
Does this work mean that the XML is the same or different from the Standard?	Partial. See comment.	The work is representative of a formal restricted subset for a given device class.
The "Replace" directive may be a challenge as it rewrites the superclass (C12.19 Section 3.66).	See comment.	This is a limitation (IEC CIM like) of this UML model. It is not an issue for the general use of UML.
Class hierarchy issue: representing the Decade in the fashion shown needs further investigation.	See comment.	This is a limitation (IEC CIM like) of this UML model, and possibly of UML, as it may not be able to represent the concept of container "packages". A C12.19 Decade is a container for defining a scope for the definitions of derived types, Tables and Procedures. Alternatively, a Decade may be expressed as a container class that contains inner classes of derived types, Tables and Procedures.
Presentation and preservation of C12.19 Section 4 (grouping, properties of tables, procedures and types) shall be maintained.	No. See comment.	This is not expressed in the UML data Meta Model as this UML does not include meta-attributes and process management attributes.
Presentation and preservation of C12.19 Annex G (references to types, constants, values, scoping rules, naming) shall be maintained.	No. See comment.	Not possible with this restriction on UML.

Table 2: Meta Modeling Issues



5 From Standard Syntax to UML

The C12.19 Standard syntax (also referred to as the “Document Form”) is captured in Microsoft Word and distributed as a printed document or in portable document format (PDF). The printed/electronic Standard contains rules to render the Standard in XML/Schema, which can then be analyzed by any number of software tools and contain a certain graphical view of the Standard. From that schema, a UML modeling tool can create the UML model, which may include another graphical view of the Standard.

The resulting UML model is discussed in this section. To perform the capture of C12.19 in UML, advantage was taken of two technologies:

- 1) The ability to generate an EDL Schema directly from the IEEE version of the standard
- 2) The ability of Sparx Systems Enterprise Architect (EA) to import any (reduced) XML Schema to produce a UML model

ANSI C12.19 is written using a “pseudo-Pascal¹⁶” textual syntax. As such, it may be not readily recognizable by contemporary software architects and developers. Additionally, the standard form is not “machine-readable” but must be read and interpreted.

Advantage has been taken, however, by recognizing the uniformity of the formatted document to “parse” the document into a machine-readable form. In the present analysis, this approach was used to generate an EDL Schema directly from the IEEE 1377-2012 version of the standard.

The C12.19 EDL Schema reduction process used in this paper included the following steps:

- Processed the IEEE Std 1377 + IEEE Std 1703 Table Definition Language to yield a composite Standards EDL schema definition, ensuring that all EDL element types are globally named.
- Recast the Standard EDL into a simplified EDL schema that did not contain any polymorphic Elements by removing all references to seldom used data types (such as BNF).
- Recast the resulting EDL schema by collapsing multiple Element names that exist in a single class hierarchy down to a single Element that obeys W3C XML type uniqueness constraints.
- Removed all element index attributes (object reference identifiers) and pending table header attributes.
- Applied CamelCase transformation to resulting schema to yield an EDL Schema that can be imported into enterprise architect and is familiar in style to persons that are familiar with UML naming conventions.

Once in XML Schema form, it becomes possible to use the import capabilities of UML model editing tools to assemble a UML model for the standard.

XML Schema and UML are extremely flexible representation technologies. With this flexibility there is more than one way the same information can be modeled and presented. This is conventionally termed modeling style. A principle purpose of the PAP06 activities was to produce a UML model that

¹⁶ Pascal is an influential imperative and procedural programming language, designed in 1968–1969 and published in 1970 by Niklaus Wirth as a small and efficient language intended to encourage good programming practices using structured programming and data structuring. [Wikipedia]



represented as accurately and completely as possible the C12.19 standard. This UML model will hopefully be readily understood by modelers and architects studying and applying technologies in the Smart Grid.

The developers of this paper tried numerous styles of exporting and importing the automatically transformed C12.19 model. Eventually, the present form was achieved. It had the following key properties:

- The model elements produced in UML corresponded in a recognizable way with the written document.
- All terms and names were converted to CamelCase¹⁷ so that the nomenclature had the look and feel of similar models.
- The model elements were concise enough to be presented in class diagrams to show the composition of the table models.
- Two styles of capture were produced – one that represents all components as classes with associations, and, one which represents major elements as classes and minor elements as attributes of classes. Both are presented in the resulting UML model.

The EDL resulting artifacts with and without CamelCase conversions are available as:

- Schema: Reference [17]
- UML Model: Reference [18]
- Style sheet for C12.19 strict to CamelCase: Reference [19]

¹⁷ CamelCase is a naming technique where multiple terms are concatenated together with the first letter of each term capitalized and the remaining letters in lower case. This representation is concise and usable in any software or design tool.



5.1 Graphical Representations

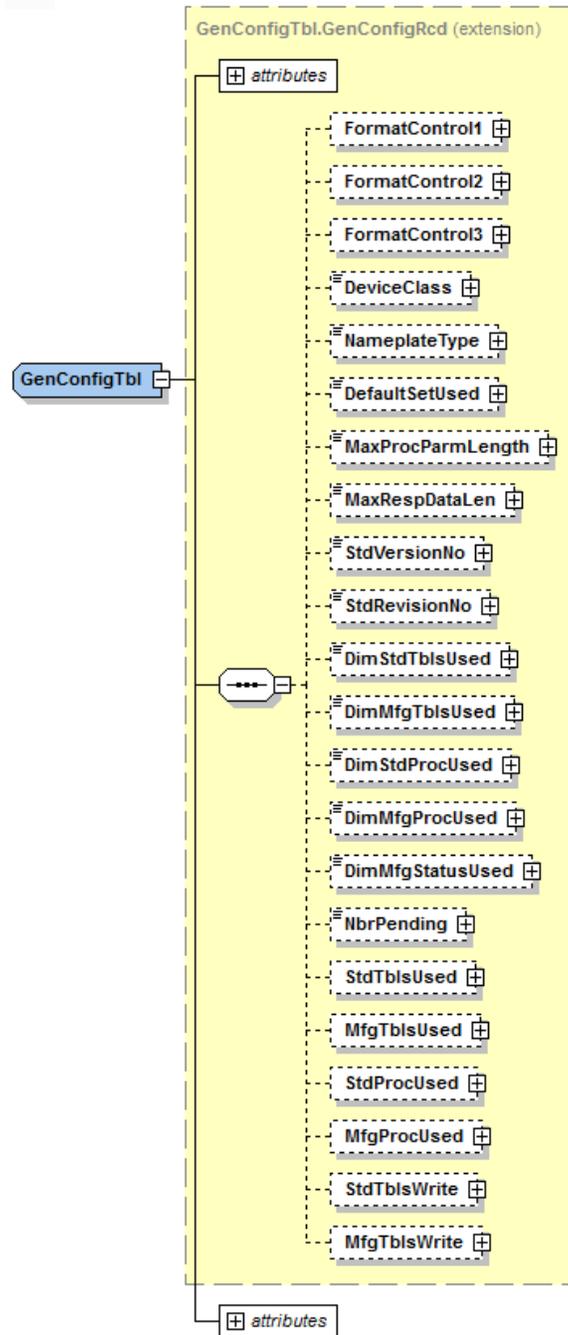


Figure 9: Table 00, GenConfigTbl

Figure 9 shows a rendering of the XML schema for Table 00, the General Configuration Table (GenConfigTbl).

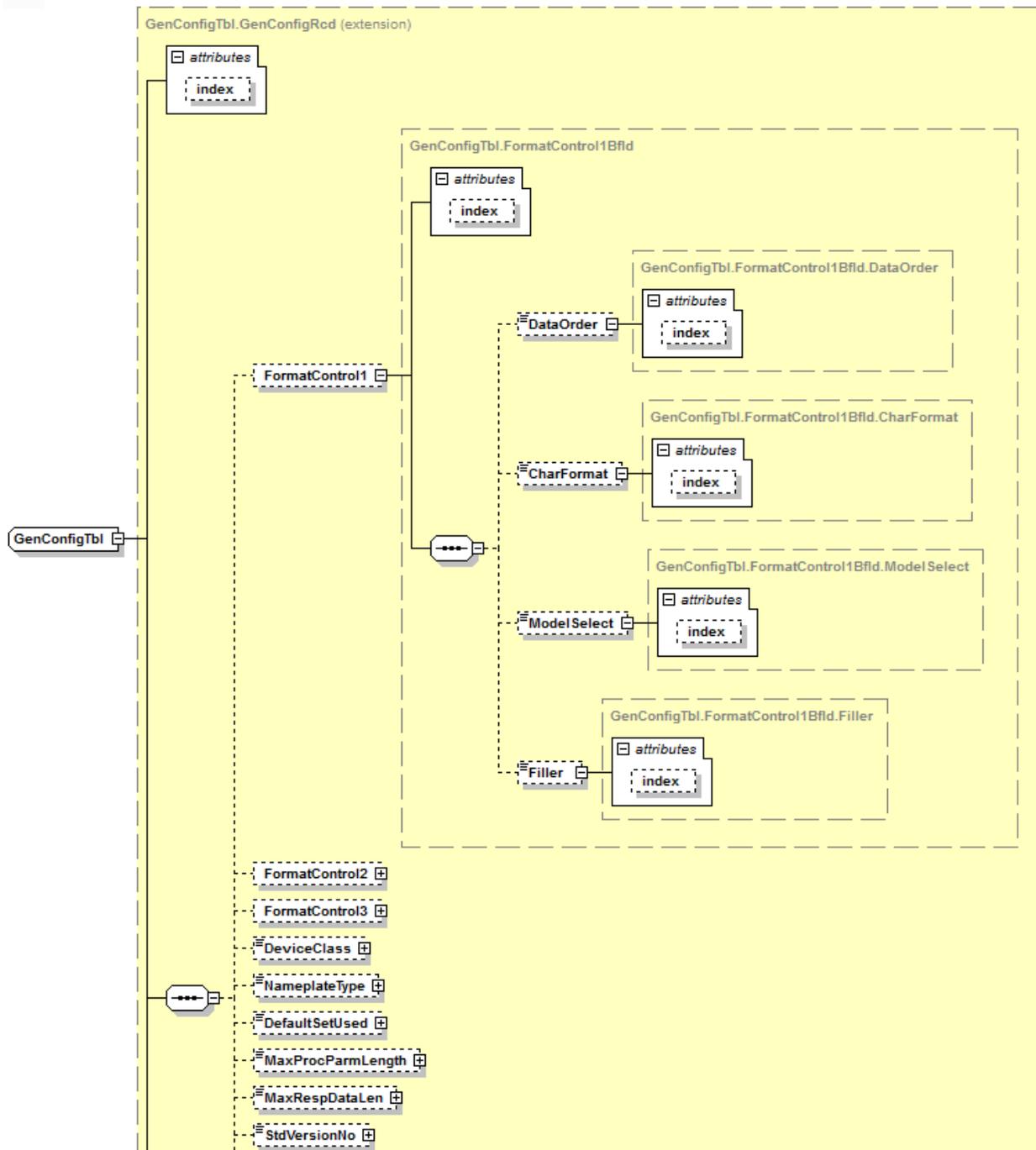


Figure 10: Expansion of `FORMAT_CONTROL_1_BFLD`

Figure 10 shows the XML expansion of the `FORMAT_CONTROL_1_BFLD`.



▲ xsd:complexType = name GenConfigTbl.FormatControl1Bfld	
▲ xsd:sequence	
▲ xsd:element	
= name	DataOrder
= type	GenConfigTbl.FormatControl1Bfld.DataOrder
= minOccurs	0
= maxOccurs	1
▲ xsd:element	
= name	CharFormat
= type	GenConfigTbl.FormatControl1Bfld.CharFormat
= minOccurs	0
= maxOccurs	1
▲ xsd:element	
= name	ModelSelect
= type	GenConfigTbl.FormatControl1Bfld.ModelSelect
= minOccurs	0
= maxOccurs	1
▲ xsd:element	
= name	Filler
= type	GenConfigTbl.FormatControl1Bfld.Filler
= minOccurs	0
= maxOccurs	1
▲ xsd:attributeGroup	
= ref	Elementindexattr
▲ xsd:complexType	
= name	GenConfigTbl.FormatControl1Bfld.DataOrder
▲ xsd:simpleContent	
▲ xsd:restriction	
= base	UInt
▲ xsd:minInclusive	
= value	0
▲ xsd:maxInclusive	
= value	1
▲ xsd:enumeration	
= value	0
▲ xsd:enumeration	
= value	1
▲ xsd:complexType	
= name	GenConfigTbl.FormatControl1Bfld.CharFormat
▲ xsd:simpleContent	
▼ xsd:restriction base=UInt	
▲ xsd:complexType	
= name	GenConfigTbl.FormatControl1Bfld.ModelSelect
▲ xsd:simpleContent	
▼ xsd:restriction base=UInt	
▲ xsd:complexType	
= name	GenConfigTbl.FormatControl1Bfld.Filler
▲ xsd:simpleContent	
▼ xsd:restriction base=Fill	
▼ xsd:complexType name=GenConfigTbl.FormatControl2Bfld	

Figure 11: Expansion of FORMAT_CONTROL_1_BFLD

Figure 11 shows the alternate composition view of the same part of the schema as Figure 10.



Figure 12: Table 00 Standard Syntax and UML

Figure 12 shows the UML rendering using attributes of Table 00 side-by-side with the Standard syntax.

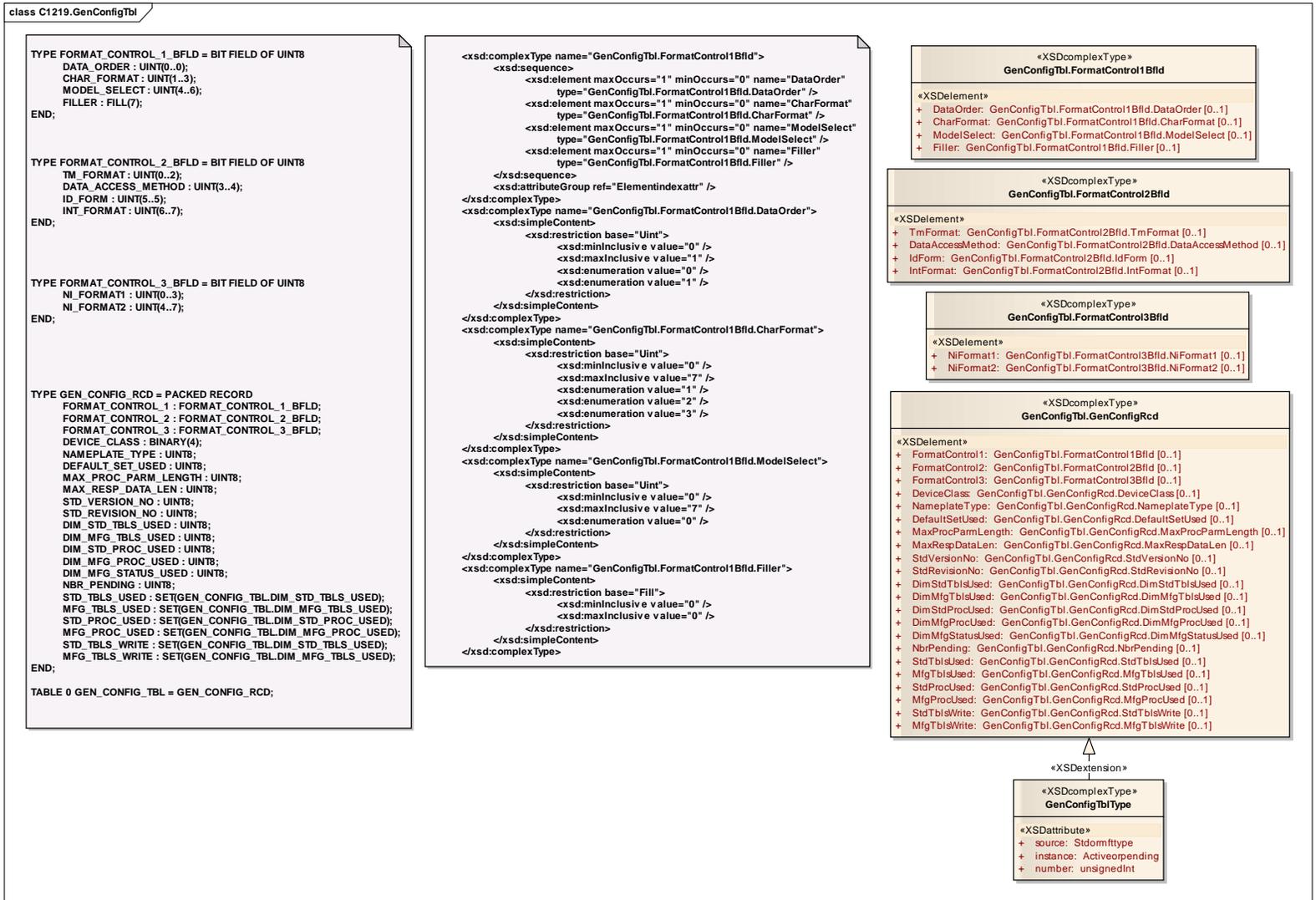


Figure 13: Relationship Between Standard Syntax, the Schema and UML

1

2

3

Figure 13 shows the progression and relationship between the Standard syntax, rendering in XML and UML.



5.2 Rendering with Attributes vs. Associations

There are many ways one can model an object or process model using UML. One of the challenges is to establish the correct relationships among the modeled objects. This needs to be done with simplicity of modeling and implementation in mind. One may create relationships among objects by creating direct “associations” among objects (e.g., when object A implements, or it is a generalization, or it is a restriction, or it is being restricted by object B). Alternatively, it is possible to create a reference to other objects through the use of “attributes” (e.g., when object A is a container, or it uses object B). The use of associations tends to create visual links among all objects that are associated. This can pose a visual challenge (is messy) when importing a large W3C schema. Use of Attributes on the other hand is less likely to create this visualization challenge, while it is also more aligned (visually) with the “Document Form” manifestation of the C12.19 Table meta-model. For these reasons, the UML diagrams in this paper (e.g., Figure 13: Relationship Between Standard Syntax, the Schema and UML) utilize the UML Attribute notation.



6 Recommendations and Conclusions

This project has successfully produced a UML model representation of the ANSI C12.19 information model. The resulting model is understandable and represents a modeling style similar to those in use in the various Smart Grid standards. The underlying abstract model, termed the Meta Model, of the standard has been captured to convey the understanding of this model and thereby to provide for the guidance in building tools to interact with it.

This finite activity brought the model to a fairly advanced stage with virtually all semantics captured and represented. However, as indicated in Section 4.2.1 Limitations of UML Representation of the C12.19 Meta Model, there are some limitations to the result. The authors believe it will be valuable to address these limitations through enhancing the model. This may be done manually where the automated tools cannot provide the capture capability.

The remaining issues identified in transforming the C12.19 model to UML are summarized in Table 2: Meta Modeling Issues.

Finally, this work does not address any comments on cyber security as they relate to the SGiP CSWG review of ANSI C12.19, nor any of the other C12 standards. The production of an alternate expression of the C12.19 standard was not generated to address cyber security and, from the viewpoint of the PAP06 Working Group, and is unrelated to that topic.



7 Acknowledgements

The primary authors of this white paper were Martin Burns (Hypertek) and Avygdor Moise (FutureDOS R&D). The primary reviewers and also contributors were Lawrence Kotewa (CNT Energy), Tom Nelson (NIST), Aaron Snyder (EnerNex) and Kostas Tolios (DTE Energy).

The PAP06 WG gratefully acknowledges the assistance of the following individuals and companies.

Name	Company
Larry Barto	Southern Company – Georgia Power
Ed Beroset	Elster Solutions
David Haynes	Aclara
Terry Penn	Southern Company – Georgia Power
Carrin Tunney	DTE Energy
Ginger Zinkowski	GE Energy



References

- [1] *American National Standard For Utility Industry End Device Data Tables*, ANSI C12.19-2008, Feb. 2009.
- [2] *IEEE Draft Standard for Utility Industry Metering Communication Protocol Application Layer (End Device Data Tables)*, IEEE P1377/D9, February 2011. Available [http://standards.ieee.org/develop/project/1377.html]
- [3] *Application integration at electric utilities - System interfaces for distribution management*, IEC 61968 Series, Various. [Online] Available: <http://www.iec.ch/>
- [4] *Communication networks and systems in substations*, IEC 61850 Series, Various. [Online] Available: <http://www.iec.ch/>
- [5] *Companion Specification for Energy Metering (COSEM)*, DLMS User Association. [Online] Available: <http://www.dlms.com/documentation/dlmsuacolouredbookspasswordprotectedarea/index.html>; Excerpt: http://dlms.com/documents/Excerpt_BB10.pdf
- [6] *Device Language Message Specification (DLMS)*, DLMS User Association. [Online] Available: <http://www.dlms.com/documentation/dlmsuacolouredbookspasswordprotectedarea/index.html>; Excerpt: http://dlms.com/documents/Excerpt_GB7.pdf
- [7] *Electricity metering - Data exchange for meter reading, tariff and load control*, IEC 62056 Series, Various. [Online] Available: <http://www.iec.ch/>
- [8] *Energy management system application program interface (EMS-API)*, IEC 61970 Series, Various. [Online] Available: <http://www.iec.ch/>
- [9] Metamodeling. (2011, April 4). In *Wikipedia, the Free Encyclopedia*. Retrieved 15:47, June 27, 2011, from <http://en.wikipedia.org/w/index.php?title=Metamodeling&oldid=422328749>
- [10] *NIST Framework and Roadmap for Smart Grid Interoperability Standards, Release 1.0*, NIST SP1108. [Online] Available: <http://www.nist.gov/smartgrid/upload/FinalSGDoc2010019-corr010411-2.pdf>
- [11] *Smart Grid/AEIC AMI Interoperability Standard Guidelines for ANSI C12.19 / IEEE 1377 / MC12.19 End Device Communications and Supporting Enterprise Devices, Networks and Related Accessories, Version 2.0*, AEIC, November 19, 2010. [Online] Available: http://www.aeic.org/meter_service/AEICSmartGridStandardv2-11-19-10.pdf
- [12] Unified Modeling Language, Object Management Group. [Online] Available: <http://www.omg.org/spec/UML/2.3/>
- [13] OMG Unified Modeling Language (OMG UML), Superstructure, V2.1.2, OMG, 2007-11-02, <http://www.omg.org/spec/UML/2.1.2/Superstructure/PDF>
- [14] M. J. Burns, View Based Architecture and UML Model Discussion, [Online] Available: <http://collaborate.nist.gov/twiki-sggrid/pub/SmartGrid/SGIPConceptualModelDevelopmentSGAC/ViewBasedArchitectureAndUML20090603.doc>
- [15] Extensible Markup Language (XML) 1.0 (Third Edition), W3C Recommendation 04 February 2004, <http://www.w3.org/TR/REC-xml>



- [16] W3C XML Schema specification (May 2001), XML Schema Part 0: Primer, <http://www.w3.org/TR/xmlschema-0>, XML Schema Part 1: Structures, <http://www.w3.org/TR/xmlschema-1>, XML Schema Part 2: Datatypes, <http://www.w3.org/TR/xmlschema-2>
- [17] EDLSchema: <http://collaborate.nist.gov/twiki-ssgrid/pub/SmartGrid/PAP06Deliverable1/www.ecmx.org.edl.IEEE.0.2.1.1.EDLSchema>
- [18] UML EA model: <http://collaborate.nist.gov/twiki-ssgrid/pub/SmartGrid/PAP06Deliverable1/C1219UMLModel20120203.eap>
- [19] CamelCase style sheet: <http://collaborate.nist.gov/twiki-ssgrid/pub/SmartGrid/PAP06Deliverable1/www.ecmx.org.edl.IEEE.0.2.1.1.EDLSchemaCamelCase.xsd>



Glossary

Where capitalized, words such as “Decade,” “Element,” and “Table” are used as defined in this Glossary. Where non-capitalized, the generally-accepted English definition is implied.

AEIC	Association of Edison Illuminating Companies, see http://www.aeic.org/
AMTIT	Advanced Metering Technology Issues Team
ANSI	American National Standards Institute
Atomic Element	A restricted subset of an Element that is the smallest component of an Element that can be transmitted as an integral number of Octets without loss of its meaning or interpretation during transmission, in accordance with Octet ordering and bit packing. [ANSI C12.19-2008]
CIM	Common Information Model. In the context of this White Paper, refers to the use of IEC 61968 and IEC 61970.
COSEM	Companion Specification for Energy Metering. The DLMS/COSEM specification specifies a data model and communication protocols for data exchange with metering equipment.
Decade	A functional grouping of Tables by application into groups of ten. The Tables are numbered “X0” through “X9”, with “X” representing the Decade number. [ANSI C12.19-1997]
Default Set	Element-values provided when FLC Tables or FLC+1 Tables of any Decade are not used by the End Device. This Standard provides Default Set Element-values for all FLC and FLC+1 Table-elements. End Device vendors may provide Default Set Element-values for any Manufacturer-defined Decades FLC and FLC+1 using EDL.
DLMS	Device Language Message Specification. The DLMS/COSEM specification specifies a data model and communication protocols for data exchange with metering equipment.
Domain	Smart Grid Conceptual Model decomposition of the electricity industry from NIST SP1108, with the divisions of Bulk Generation, Transmission, Distribution, Customer, Operations, Markets and Service Provider.
EDL	Exchange Data Language is for encoding, import and export of Default Sets defined by this Standard, Table Element values retrieved from an End Device, Table Element values to be stored in an End Device, Manufacturer constant Element-values and Utility or Customer site-specific constant Element-values that need to be known by application. The Exchange Data Language Element-values are encoded using XML. [ANSI C12.19-2008]



Element	The union of all of the Atomic Elements which share the same index prefix. An Element can be any type, a set, an array or a selection from an array. [ANSI C12.19-2008]
End Device	The closest device to the sensor or control point within a metering application communication system which is compliant with the Utility Industry End Device Data Tables. [ANSI C12.19-1997]
Head-end system	The head-end system receives the stream of meter data brought back to the utility by an AMR/AMI system. Head-end systems may perform a limited amount of data validation before either making the data available for other systems to request or pushing the data out to other systems. [Adapted from http://sites.energetics.com/MADRI/pdfs/tech_def2.doc].
IEC	International Electrotechnical Commission, see http://www.iec.ch/
IEEE	Institute of Electrical and Electronics Engineers, see http://www.ieee.org/
IKB	Interoperability Knowledge Base. One example is found at http://collaborate.nist.gov/twiki-sggrid/bin/view/SmartGrid/InteroperabilityKnowledgeBase
MDMS	Meter data management system. Software and hardware that minimally allows for validation, estimation and editing of meter data.
MOU	Memorandum of Understanding
NEMA	The Association of Electrical Equipment and Medical Imaging Manufacturers, see http://www.nema.org/
NIST	National Institute of Standards and Technology, see http://www.nist.gov/
Octet	A sequence of eight bits. [ANSI C12.19-1997]
PAP	Priority Action Plan, see http://collaborate.nist.gov/twiki-sggrid/bin/view/SmartGrid/PriorityActionPlans
SCC	Standards Coordinating Committee, see https://standards.ieee.org/about/sasb/scc.html
SGIP	Smart Grid Interoperability Panel, see http://collaborate.nist.gov/twiki-sggrid/bin/view/SmartGrid/WebHome
Table	Functionally related utility application data Elements, grouped together into a single data structure for transport. [ANSI C12.19-1997]
TDL	Syntax used for detailing the Standard's or an End Device's Table structure, defined types and constraints in a machine-readable format. The Table Definition Language is encoded using XML. [ANSI C12.18-2008]
UML	Unified Modeling Language, see http://www.uml.org/
W3C	World Wide Web Consortium, see http://www.w3c.org/



- XMI eXtensible Markup Language Metadata Interchange, see <http://www.omg.org/spec/XMI/>
- XML Extensible Markup Language, see <http://www.w3.org/XML/>



Useful Files

The following links are for the artifacts described in this white paper:

EDL Schema File: [http://collaborate.nist.gov/twiki-
sggrid/pub/SmartGrid/PAP06Deliverable1/www.ecmx.org.edl.IEEE.0.2.1.1.EDLSchema](http://collaborate.nist.gov/twiki-
sggrid/pub/SmartGrid/PAP06Deliverable1/www.ecmx.org.edl.IEEE.0.2.1.1.EDLSchema)

EAP File: [http://collaborate.nist.gov/twiki-
sggrid/pub/SmartGrid/PAP06Deliverable1/C1219UMLModel20120203.eap](http://collaborate.nist.gov/twiki-
sggrid/pub/SmartGrid/PAP06Deliverable1/C1219UMLModel20120203.eap)

CamelCase style sheet: [http://collaborate.nist.gov/twiki-
sggrid/pub/SmartGrid/PAP06Deliverable1/C1219EDL2CamelCase.xslt](http://collaborate.nist.gov/twiki-
sggrid/pub/SmartGrid/PAP06Deliverable1/C1219EDL2CamelCase.xslt)