IEC 61968-11

Edition 2.0 2013-03

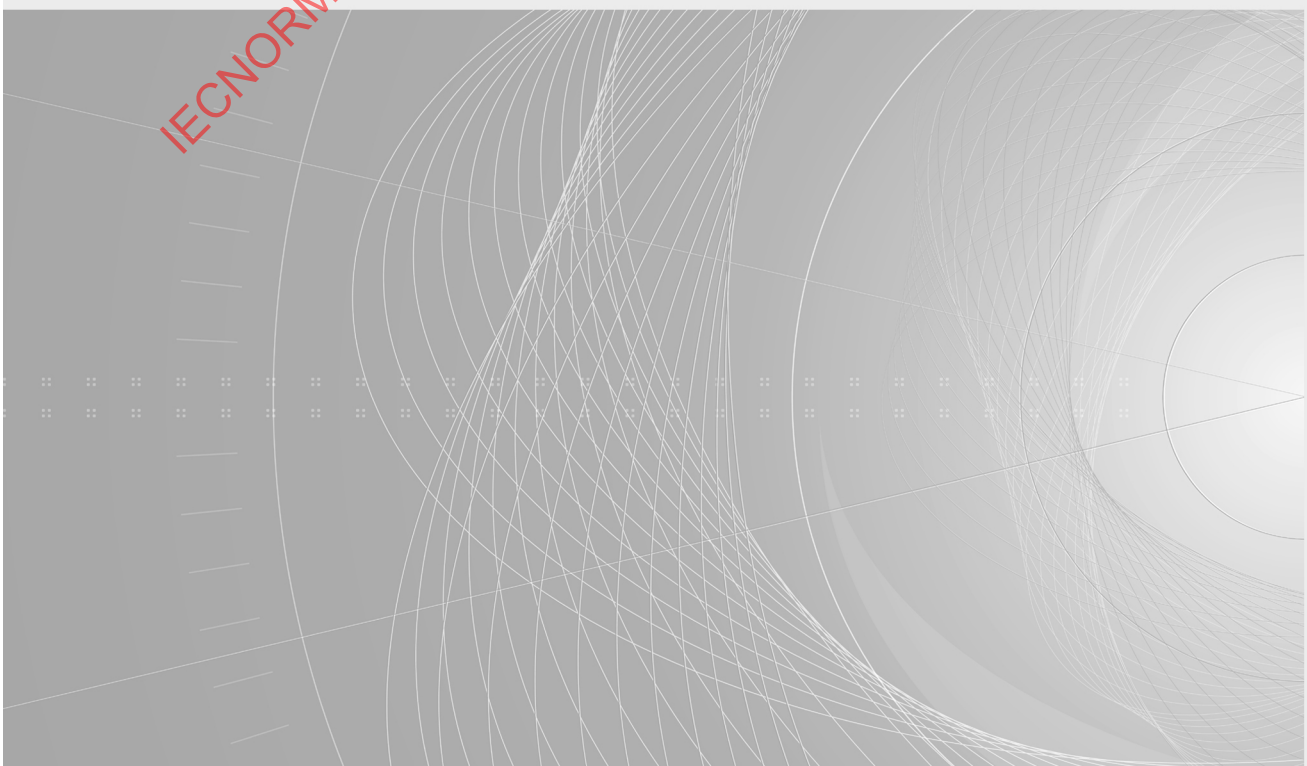
INTERNATIONAL STANDARD

NORME INTERNATIONALE

colour inside

Application integration at electric utilities – System interfaces for distribution management –
Part 11: Common information model (CIM) extensions for distribution

Intégration d'applications pour les services électriques – Interfaces système pour la gestion de distribution –
Partie 11: Extensions du modèle d'information commun (CIM) pour la distribution



IEC 61968-11:2013

IEC Central Office
3, rue de Varembé
CH-1211 Geneva 20
Switzerland

Tel.: +41 22 919 02 11
Fax: +41 22 919 03 00
info@iec.ch
www.iec.ch

About the IEC

The International Electrotechnical Commission (IEC) is the leading global organization that prepares and publishes International Standards for all electrical, electronic and related technologies.

About IEC publications

The technical content of IEC publications is kept under constant review by the IEC. Please make sure that you have the latest edition, a corrigenda or an amendment might have been published.

Useful links:

IEC publications search - www.iec.ch/searchpub

The advanced search enables you to find IEC publications by a variety of criteria (reference number, text, technical committee,…).
It also gives information on projects, replaced and withdrawn publications.

IEC Just Published - webstore.iec.ch/justpublished

Stay up to date on all new IEC publications. Just Published details all new publications released. Available on-line and also once a month by email.

Electropedia - www.electropedia.org

The world's leading online dictionary of electronic and electrical terms containing more than 30 000 terms and definitions in English and French, with equivalent terms in additional languages. Also known as the International Electrotechnical Vocabulary (IEV) on-line.

Customer Service Centre - webstore.iec.ch/csc

If you wish to give us your feedback on this publication or need further assistance, please contact the Customer Service Centre: csc@iec.ch.

A propos de la CEI

La Commission Electrotechnique Internationale (CEI) est la première organisation mondiale qui élabore et publie des Normes internationales pour tout ce qui a trait à l'électricité, à l'électronique et aux technologies apparentées.

A propos des publications CEI

Le contenu technique des publications de la CEI est constamment revu. Veuillez vous assurer que vous possédez l'édition la plus récente, un corrigendum ou amendement peut avoir été publié.

Liens utiles:

Recherche de publications CEI - www.iec.ch/searchpub

La recherche avancée vous permet de trouver des publications CEI en utilisant différents critères (numéro de référence, texte, comité d'études,…).
Elle donne aussi des informations sur les projets et les publications remplacées ou retirées.

Just Published CEI - webstore.iec.ch/justpublished

Restez informé sur les nouvelles publications de la CEI. Just Published détaille les nouvelles publications parues. Disponible en ligne et aussi une fois par mois par email.

Electropedia - www.electropedia.org

Le premier dictionnaire en ligne au monde de termes électroniques et électriques. Il contient plus de 30 000 termes et définitions en anglais et en français, ainsi que les termes équivalents dans les langues additionnelles. Egalement appelé Vocabulaire Electrotechnique International (VEI) en ligne.

Service Clients - webstore.iec.ch/csc

Si vous désirez nous donner des commentaires sur cette publication ou si vous avez des questions contactez-nous: csc@iec.ch.

IEC 61968-11

Edition 2.0 2013-03

INTERNATIONAL STANDARD

NORME INTERNATIONALE

colour inside

Application integration at electric utilities – System interfaces for distribution management – Part 11: Common information model (CIM) extensions for distribution

Intégration d'applications pour les services électriques – Interfaces système pour la gestion de distribution – Partie 11: Extensions du modèle d'information commun (CIM) pour la distribution

INTERNATIONAL ELECTROTECHNICAL COMMISSION

COMMISSION ELECTROTECHNIQUE INTERNATIONALE

PRICE CODE XH
CODE PRIX

ICS 33.200

ISBN 978-2-83220-662-1

CONTENTS

INTERNATIONAL ELECTROTECHNICAL COMMISSION

APPLICATION INTEGRATION AT ELECTRIC UTILITIES – SYSTEM INTERFACES FOR DISTRIBUTION MANAGEMENT –

Part 11: Common information model (CIM) extensions for distribution

FOREWORD

1) The International Electrotechnical Commission (IEC) is a worldwide organization for standardization comprising all national electrotechnical committees (IEC National Committees). The object of IEC is to promote international co-operation on all questions concerning standardization in the electrical and electronic fields. To this end and in addition to other activities, IEC publishes International Standards, Technical Specifications, Technical Reports, Publicly Available Specifications (PAS) and Guides (hereafter referred to as "IEC Publication(s)"). Their preparation is entrusted to technical committees; any IEC National Committee interested in the subject dealt with may participate in this preparatory work. International, governmental and non-governmental organizations liaising with the IEC also participate in this preparation. IEC collaborates closely with the International Organization for Standardization (ISO) in accordance with conditions determined by agreement between the two organizations.
2) The formal decisions or agreements of IEC on technical matters express, as nearly as possible, an international consensus of opinion on the relevant subjects since each technical committee has representation from all interested IEC National Committees.
3) IEC Publications have the form of recommendations for international use and are accepted by IEC National Committees in that sense. While all reasonable efforts are made to ensure that the technical content of IEC Publications is accurate, IEC cannot be held responsible for the way in which they are used or for any misinterpretation by any end user.
4) In order to promote international uniformity, IEC National Committees undertake to apply IEC Publications transparently to the maximum extent possible in their national and regional publications. Any divergence between any IEC Publication and the corresponding national or regional publication shall be clearly indicated in the latter.
5) IEC itself does not provide any attestation of conformity. Independent certification bodies provide conformity assessment services and, in some areas, access to IEC marks of conformity. IEC is not responsible for any services carried out by independent certification bodies.
6) All users should ensure that they have the latest edition of this publication.
7) No liability shall attach to IEC or its directors, employees, servants or agents including individual experts and members of its technical committees and IEC National Committees for any personal injury, property damage or other damage of any nature whatsoever, whether direct or indirect, or for costs (including legal fees) and expenses arising out of the publication, use of, or reliance upon, this IEC Publication or any other IEC Publications.
8) Attention is drawn to the Normative references cited in this publication. Use of the referenced publications is indispensable for the correct application of this publication.
9) Attention is drawn to the possibility that some of the elements of this IEC Publication may be the subject of patent rights. IEC shall not be held responsible for identifying any or all such patent rights.

International Standard IEC 61968-11 has been prepared by IEC technical committee 57: Power systems management and associated information exchange.

The text of this standard is based on the following documents:

FDIS	Report on voting
57/1295/FDIS	57/1326/RVD

Full information on the voting for the approval of this standard can be found in the report on voting indicated in the above table.

This publication has been drafted in accordance with the ISO/IEC Directives, Part 2.

This second edition cancels and replaces the first edition published in 2010. It constitutes a technical revision.

Major changes with respect to the first edition are summarised below[1];

- Introduction of new classes to support flexible naming of identified objects (new classes available in base CIM, IEC 61970-301).
- Introduction of new classes to support single line diagrams exchange (new classes available in base CIM, IEC 61970-301).
- Consolidated transmission and distribution models for lines, transformers, switching, sensing and other auxiliary equipment (some Ed.1 classes slightly changed and moved from DCIM to base CIM, IEC 61970-301, other new classes available in base CIM, IEC 61970-301).
- Support for separate phase definitions, typically needed for unbalanced network modelling (new classes available in base CIM, IEC 61970-301).
- Support for temporary network changes through models of cuts, jumpers and clamps (new classes available in base CIM, IEC 61970-301).
- Flexible model for organisations and their roles.
- Support for coordinate systems in description of geographical locations.
- Support for configuration events tracking.
- Lightweight model for assets and asset catalogues.
- Support for linkage between network-oriented models and premises-oriented (metering) models.
- Support for premises area network devices.

In informative sections of this document, words printed in Arial Black apply to terms that are used as tokens in the normative clauses (to facilitate the reading and the text search).

A list of all parts of the IEC 61968 series, under the general title: *Application integration at electric utilities – System interfaces for distribution management* can be found on the IEC website.

The committee has decided that the contents of this publication will remain unchanged until the stability date indicated on the IEC web site under "http://webstore.iec.ch" in the data related to the specific publication. At this date, the publication will be

- reconfirmed,
- withdrawn,
- replaced by a revised edition, or
- amended.

IMPORTANT – The 'colour inside' logo on the cover page of this publication indicates that it contains colours which are considered to be useful for the correct understanding of its contents. Users should therefore print this document using a colour printer.

[1] For enhancements in the base CIM, see IEC 61970-301 documenting CIM15.

INTRODUCTION

The IEC 61968 series of standards is intended to facilitate inter-application integration as opposed to intra-application integration. Intra-application integration is aimed at programs in the same application system, usually communicating with each other using middleware that is embedded in their underlying runtime environment, and tends to be optimised for close, real-time, synchronous connections and interactive request/reply or conversation communication models. Therefore, these inter-application interface standards are relevant to loosely coupled applications with more heterogeneity in languages, operating systems, protocols and management tools. This series of standards is intended to support applications that need to exchange data every few seconds, minutes, or hours rather than waiting for a nightly batch run. This series of standards, which are intended to be implemented with middleware services that exchange messages among applications, will complement, not replace utility data warehouses, database gateways, and operational stores.

As used in IEC 61968, a distribution management system (DMS) consists of various distributed application components for the utility to manage electrical distribution networks. These capabilities include monitoring and control of equipment for power delivery, management processes to ensure system reliability, voltage management, demand-side management, outage management, work management, automated mapping and facilities management. Standard interfaces are defined for each class of applications identified in the interface reference model (IRM), which is described in IEC 61968-1.

APPLICATION INTEGRATION AT ELECTRIC UTILITIES – SYSTEM INTERFACES FOR DISTRIBUTION MANAGEMENT –

Part 11: Common information model (CIM) extensions for distribution

1 Scope

This part of IEC 61968 specifies the distribution extensions of the common information model (CIM) specified in IEC 61970-301. It defines a standard set of extensions of common information model (CIM), which support message definitions in IEC 61968-3 to IEC 61968-9, IEC 61968-13 and IEC 61968-14[2]. The scope of this standard is the information model that extends the base CIM for the needs of distribution networks, as well as for integration with enterprise-wide information systems typically used within electrical utilities. The information model is defined in UML which is platform-independent and electronically processable language that is then used to create message payload definitions in different required formats. In this way, this standard will not be impacted by the specification, development and/or deployment of next generation infrastructures, either through the use of standards or proprietary means.

For the purposes of this part of IEC 61968, the distribution CIM (DCIM) model refers to the IEC CIM model as defined by IEC 61970-301 and this part of IEC 61968.

The common information model (CIM) is an abstract model of the major objects in an electric utility enterprise typically involved in utility operations. By providing a standard way of representing power system resources as object classes and attributes, along with their relationships, the CIM facilitates the integration of software applications developed independently by different vendors. The CIM facilitates integration by defining a common language (i.e., semantics and syntax) based on the CIM to enable these applications or systems to access public data and exchange information independent of how such information is represented internally.

IEC 61970-301 defines a core CIM for energy management system (EMS) applications, including many classes that would be useful in a wider variety of applications. Due to its size, the CIM classes are grouped into logical packages, and collections of these packages are maintained as separate International Standards. This document extends the core CIM with packages that focus on distribution management systems (DMS) including assets, work, customers, load control, metering, and others. IEC 62325-301[3] extends the CIM with packages that focus on market operations and market management applications. Other CIM extensions may be published as International Standards, each maintained by a separate group of domain experts. Depending on a project's needs, the integration of applications may require classes and packages from one or more of the CIM standards.

2 Normative references

The following documents, in whole or in part, are normatively referenced in this document and are indispensable for its application. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

IEC 60076-1, *Power transformers – Part 1: General*

---

2 IEC 61968-5, IEC 61968-6, IEC 61968-7, IEC 61968-8 and IEC 61968-14 are under consideration.

3 Under consideration.

IEC 61968-1, *Application integration at electric utilities – System interfaces for distribution management – Part 1: Interface architecture and general requirements*

IEC 61968-2, *Application integration at electric utilities – System interfaces for distribution management – Part 2: Glossary*

IEC 61970-301, *Energy management system application program interface (EMS-API) –Part 301: Common information model (CIM) base*[4]

IEC 61970-501, *Energy management system application program interface (EMS-API) – Part 501: Common Information Model Resource Description Framework (CIM RDF) schema*

IEC 62361-100, *Naming and Design Rules for CIM Profiles to XML Schema Mapping*[5]

IEEE 802.3, *IEEE Standard for Information technology-Specific requirements – Part 3: Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications*

3 Terms and definitions

For the purposes of this document, the terms and definitions given in IEC 61968-2 and the following apply.

NOTE Refer to International Electrotechnical Vocabulary, IEC 60050, for general glossary definitions.

3.1
energy management system
EMS
computer system comprising a software platform providing basic support services and a set of applications providing the functionality needed for the effective operation of electrical generation and transmission facilities so as to assure adequate security of energy supply at minimum cost

3.2
distribution management system
DMS
computer system comprising a software platform providing basic support services and a set of applications providing the functionality needed for the effective operation of electrical distribution facilities so as to assure adequate security of energy supply at minimum cost

3.3
unified modeling language
UML
formal and comprehensive descriptive language with diagramming techniques used to represent software systems, from requirements analysis, through design and implementation, to documentation

Note 1 to entry: UML has evolved from a collection of methods contributed by different practitioners, into an International Standard. The CIM relies on UML for defining the model, and automated tools generate the documentation, schemas, and other artefacts directly from the UML. A basic understanding of UML is necessary to understand the CIM.

4 5th edition. Under consideration.

5 Under consideration.

3.4
common information model with distribution extensions
DCIM
union of the base CIM in IEC 61970-301 and additional packages defined in this document, IEC 61968-11

Note 1 to entry: The DCIM is intended to address most of the domain modelling needs of a DMS; however, a specific project may require other CIM packages or extensions.

3.5
profile
subset of DCIM classes, associations and attributes needed to accomplish a specific type of interface

Note 1 to entry: It may be expressed in XSD, RDF, and/or OWL files. A profile can be tested between applications. A profile is necessary in order to "use" the DCIM. Several profiles are defined in other parts of the IEC 61968 family of standards.

3.6
XML schema
schema used to define the structure, content, and semantics of extensible mark-up language (XML) files

Note 1 to entry: XML schemas are generally found in files with an "xsd" extension. The DCIM uses XSD files to define inter-application messages in most domain areas, except for power system model exchange.

3.7
resource description framework
RDF
web (W3C) standard used to represent information models

Note 1 to entry: RDF is more powerful than XSD because it can describe a data model, not just an XML file. The DCIM uses a subset of RDF to support power system model exchange.

3.8
web ontology language
OWL
another Web (W3C) standard that includes RDF and extends it

Note 1 to entry: OWL is more powerful than RDF in supporting data types, enumerations, more details of class relationships and associations, etc. Future DCIM profiles may use OWL.

4 CIM specification

4.1 CIM modelling notation

The CIM is defined using object-oriented modelling techniques. Specifically, the CIM specification uses the Unified modeling language (UML) notation, which defines the CIM as a group of packages.

Each package in the CIM contains one or more class diagrams showing graphically all the classes in that package and their relationships. Each class is then defined in text in terms of its attributes and relationships to other classes.

The UML notation is described in Object Management Group (OMG) documents and several published textbooks.

4.2 CIM packages

4.2.1 General

The CIM is partitioned into a set of packages. A package is a general purpose means of grouping related model elements. The packages have been chosen to make the model easier to design, understand and review. The common information model consists of several sets of packages. Entities may have associations that cross many package boundaries. Each application will use information represented in several packages.

4.2.2 CIM packages

The comprehensive CIM is partitioned into groups of packages for convenience. The groups comprising DCIM are:

- IEC 61970-301 (base CIM, defining data types and power system resources as required by typical EMS and DMS control centre applications);
- this part of IEC 61968.

Figure 1 shows the relevant[6] currently defined CIM normative packages and their dependency relationships. The dashed line indicates a dependency relationship, with the arrowhead pointing from the dependent package to the package on which it has a dependency.

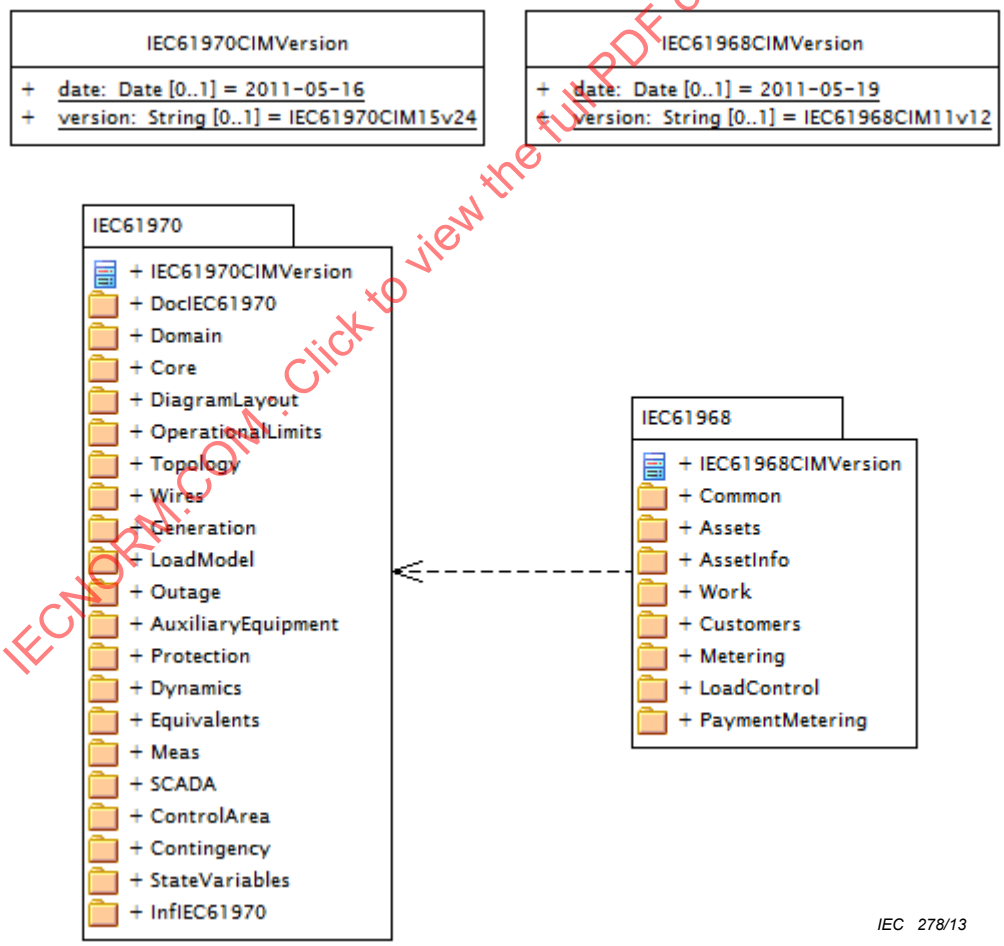


Figure 1 – CIM packages

6 CIM extensions for markets, IEC 62325-301 are also CIM normative packages, but not used in DCIM and thus not displayed.

NOTE The contents of the base CIM referred to from within this part of IEC 61968 were auto-generated from the base CIM UML electronic model release IEC61970CIM15v31.

4.2.3 CIM extensions for distribution packages (this part of IEC 61968)

The base CIM model as defined by IEC 61970-301 defines a set of sub-packages which includes **Domain**, **Wires**, **AuxiliaryEquipment**, **Topology**, **Measurements**, **Equivalents** and **Core**, as well as several other ones. IEC 61968-3 to IEC 61968-9 and IEC 61968-13 required extensions to the CIM model as specified by IEC 61970-301 in order to describe the objects and associated properties which are relevant to distribution modelling and information exchanges applicable not only to typical control room systems, but also with enterprise and partner systems, as well as metering systems and devices. Therefore, just as applications in the distribution domain use classes from the base CIM, so might applications outside the distribution domain use classes defined in this document (for instance, market extensions in IEC 62325-301).

Figure 2 shows the packages defined for IEC 61968-11 CIM extensions for distribution. Notes on the left hand side of the figure indicate the part of IEC 61968 that has been driving the definition of classes within the respective package. Note, however, that different documents in the IEC 61968 series, as well as different applications that use CIM for information exchange will typically define message payloads using classes from several packages, including some defined outside of this document.

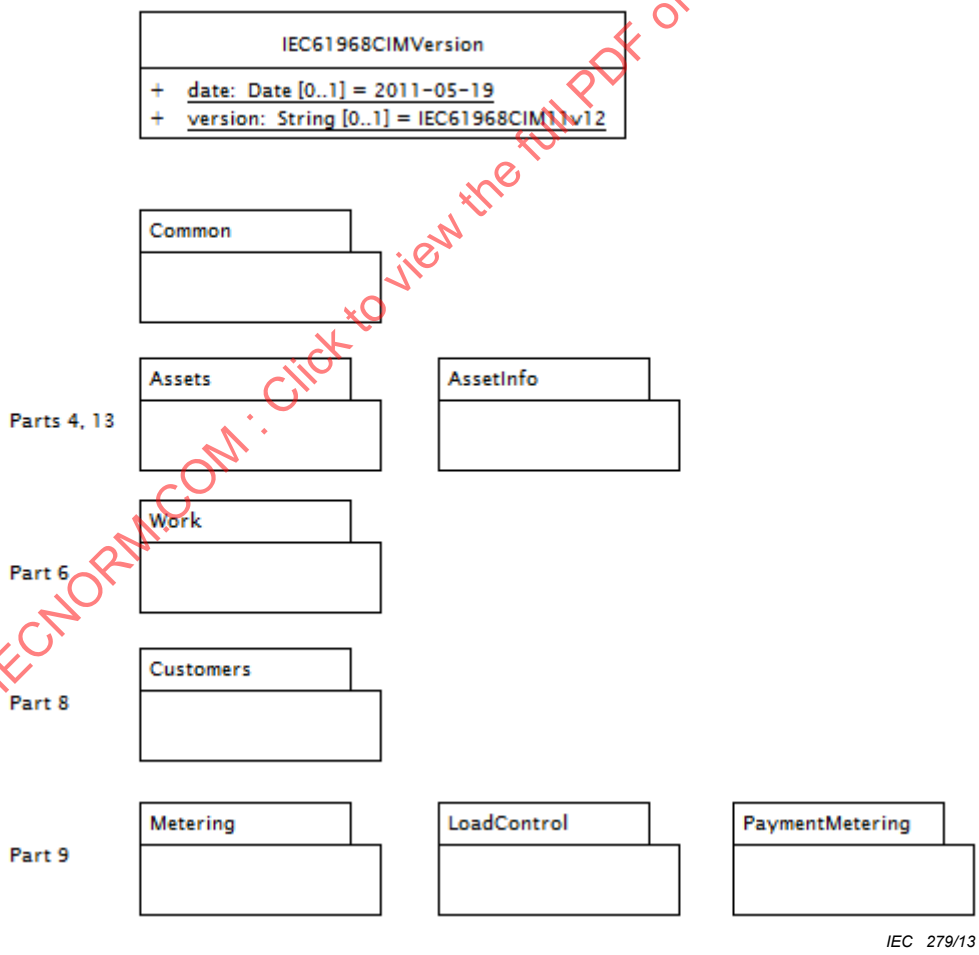


Figure 2 – CIM extensions for distribution (DCIM) top-level packages

Clause 6 contains the specification for each of the distribution CIM packages.

NOTE The contents of the CIM defined in this part of IEC 61968 were auto-generated from the CIM UML electronic model release IEC61968CIM11v13.

4.3 CIM UML modelling

4.3.1 General

The CIM model is defined and maintained using UML. The source and point of maintenance for the CIM model is currently an Enterprise Architect (EA) file. This permits the model to be viewed and maintained graphically. The same tool is used to generate web pages which can be viewed over the internet. From the EA file, XMI file is also generated that can be used within various tools that support generation of context specific message payloads in XSD, RDF, or OWL format for IEC 61968-3 to IEC 61968-9 and IEC 61968-13.

The purpose of 4.3 is to define some principles with respect to the IEC 61968 information model and associated information exchanges. For description of different UML constructs used in the CIM model, refer to 4.3 of IEC 61970-301:—, CIM classes and relationships.

4.3.2 Scope of UML model

It is not the intent of this part of IEC 61968 and associated models to define models which satisfy all information requirements, as this would be an impossible task. The standard model is a canonical data model for enterprise systems integration and thus needs to satisfy requirements for information exchanges among different systems, i.e., message payloads defined in IEC 61968-3 to IEC 61968-9 and IEC 61968-13. Custom extensions are the matter of non-standard projects and products. They can be used to leverage CIM for information models of specific applications or systems, or for non-standard integration needs. However, custom extensions are not maintained by IEC.

The overall DCIM model has been evolving during many years, and has been published as an IEC standard only in 2010. Since that first edition, the UML model has been split into normative and informative classes and their relationships. Only normative classes, with their attributes and normative relationships, are fully documented in Clause 6. At the time of editing, the normative classes and relationships are those required for IEC 61968-4, IEC 61968-9 and IEC 61968-13[7]. They are considered stable and are expected to change little.

In contrast, informative classes and their informative relationships are not documented in this specification. They are present in the electronic UML model and will be promoted to normative classes stepwise, with the new editions of IEC 61968-3 to IEC 61968-9 and IEC 61968-13. Some of those classes will be kept informative as long as they are considered unstable and likely to change, and others might be removed because they do not participate in standard information exchange.

NOTE The next anticipated set of classes that is to be promoted from informative to normative in DCIM 12 for the next edition of IEC 61968-11 are those needed to support the maintenance cycle of IEC 61968-3, IEC 61968-4, IEC 61968-9 and IEC 61968-13, and IEC 61968-6 (1st edition) and IEC 61968-8 (1st edition).

4.3.3 Extensibility

It is fully the intent to permit extensions to the information exchange model. Extensions should utilize a local namespace. The namespace shall also serve to identify the origination of the class or property.

An extension can be defined in a UML model and/or a physical model such as XML schema:

- If an extension is made in UML model, such extension should be made in a different package as a separate model layer or context rather than be added or modified into CIM model directly. A “targetNamespace” tagged value may be used for this purpose if such extension is meant for an XSD generation.

---

7 The only profile defined in the draft 2nd edition of IEC 61968-3 as of this writing is fully supported by base CIM classes, IEC 61970-301.

- If an extension is made for XSD only, a target namespace has to be different than a CIM XSD namespace to identify the origin of the extension.

The extension namespace may follow a naming convention: http://<organization>/<version control>/<profile>.

4.3.4 Message payload (profile) definition

4.3.4.1 General

A profile is a restricted subset of DCIM classes, associations and attributes needed to accomplish a specific type of interface. A profile defines the message payload from CIM model (IEC 61968-11 and IEC 61970-301) and the header format (from implementation profiles IEC 61968-100 or IEC 61970-552) and information to exchange (specific to profile document in IEC 61968-3 to IEC 61968-9 and IEC 61968-13).

One way of defining standard message payloads (profiles) is by using applications such as various available open source tools providing support for CIM-based profile creation. Other methods and tools may be used, including hand coding.

Currently, two grammars of XML are used for DCIM-based profile definition in IEC 61968 series.

4.3.4.2 XML schema syntax

IEC 61968-3 to IEC 61968-9 define profiles in XML syntax according to IEC 62361-100. These profiles specify the format and content of the DCIM-based payload part of the message that gets exchanged in various integration scenarios, as defined in IEC 61968-100. Message headers use combinations of nouns and verbs. The nouns usually refer to the concepts defined within the DCIM model (payload), and the verbs are in support of transport technology.

The verb usually indicates which attributes are required. In the case of **create/created** verbs, typically all attributes defined in a profile are required, while with **get**, **cancel/cancelled**, **delete/deleted** and **close/closed** verbs only object identifiers are typically required. The **change/changed** verb would require an object identifier and the values of attributes to be changed.

The recommendations for defining DCIM-based profiles for data exchange are given in IEC 61968-1.

The verbs defined for use within IEC 61968 compliant interfaces, as well as message envelope, are specified in IEC 61968-100.

The nouns defined for use within IEC 61968 compliant interfaces are specified in IEC 61968-3 to IEC 61968-9.

4.3.4.3 RDF schema syntax

IEC 61970-501 defines a subset of RDF schema that is used for bulk network model exchanges. The profile format and header information is specified in IEC 61970-552[8], and is currently used by the profiles defined in IEC 61968-4 and IEC 61968-13.

8 Under consideration.

4.3.4.4 Data exchange contexts supported by DCIM

DCIM-based profiles defined in IEC 61968 cover two major data exchange contexts among enterprise systems:

- Bulk data exchanges, for the purpose of configuration of systems. This kind of business process is also known as model management, master data management, or data engineering. It requires message payloads that allow for establishing and maintaining the relationships between instances of various business entities common to two or more enterprise systems.
- Operational (dynamic) data exchanges among running systems. The data exchanged over this kind of interface assumes previous configuration of business entities in various systems to their common, DCIM-based representation at the interface, but not necessarily within the system.

As of this writing, profiles in RDFS and OWL syntax are mainly used for bulk, file-based data exchanges of large data sets, while XSD profiles are used in both bulk and operational message payloads.

4.4 DCIM model concepts and examples

4.4.1 General

4.4 describes some examples of modelling in the distribution domain and general electrical utility enterprise with DCIM. Some concepts are complementary to the base CIM examples presented in IEC 61970-301 (such as network modelling typical for distribution networks). Other concepts are general for any electrical utility enterprise (such as locations, documents, organisations), or specific to distribution domain (e.g., metering and customers).

4.4.2 Common concepts

4.4.2.1 Key classes in DCIM

Base CIM in IEC 61970-301 mainly defines the function of electrical network elements through **PowerSystemResource** class and its specialisations, for the needs of information exchange in the context of control centre applications and systems. DCIM in this part of IEC 61968 adds some key classes to support (a) physical description of those network elements, (b) the whole model for metering domain, as well as (c) information exchange related to network operations and planning and meter management and control, in the context of the whole utility enterprise (including outage, customers and work management).

Figure 3 shows key classes defined in the DCIM. Relationships between them, if existing, are not shown in this figure, but rather in the following subclauses. In fact, there are relatively few direct relationships between these key DCIM classes, in favour of more specific relationships among their specialisations. The UML diagram in Figure 3 shows that all the key classes are identified objects, i.e., the name of their superclass, **IdentifiedObject**, is displayed in the upper right corner of a class.

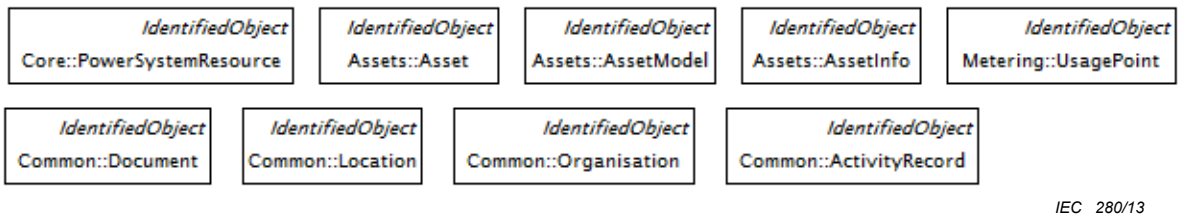


Figure 3 – DCIM key classes

An **Asset** is a tangible resource of the utility, including power system equipment, end devices, vehicles, tools, cabinets, buildings, etc. For electrical network equipment, the function of the asset is modelled with the **PowerSystemResource** hierarchy, defined fully[9] in the wires model (refer to IEC 61970-301 and model package **IEC61970::Wires**). Asset description places emphasis on the physical characteristics and the lifecycle of the equipment performing that function, and relies on **PowerSystemResource** hierarchy for electrical connectivity.

AssetModel provides details about the **Asset** as a product of a given manufacturer, so one **AssetModel** and its associated **AssetInfo** can describe many instances of **Asset**.

AssetInfo is the container for datasheet information about **Asset** and **AssetModel**, and it can also be referenced by multiple instances of **Asset** and **PowerSystemResource** (i.e., shared).

A **Location** is the place, scene, or point of something where someone or something has been, is, and/or will be at a given moment in time. It is defined with one or more position points (coordinates) in a given coordinate system.

A **UsagePoint** is the logical or physical point in the network to which meter readings or events may be attributed. It is used at the place where a physical and virtual meter may be located. This class provides the link between the network-oriented model of electrically connected equipment (**PowerSystemResource** specialisations) and the asset and premises-oriented model of the metering domain.

A **Document** is a grouping of information collected, often managed as a part of a business process. It will frequently contain references to other objects, such as assets, persons and power system resources. **Organisations** are business entities that may have roles as utilities, contractors, suppliers, manufacturers, customers, tax authorities, etc.

ActivityRecord records activity for an entity at a point in time. Activity may be for an event that has already occurred or for a planned activity.

Following subclauses discuss in some more detail the key DCIM classes, their specialisations and relationships, and indicate their intended usage.

4.4.2.2 Locations and geographical representations

To allow for locating different entities and activities in space, DCIM provides the class **Location**. Entities and activities can be located by their different kinds of addresses as well as by position coordinates, **PositionPoint** in a given **CoordinateSystem**. Changes in configuration of a **Location** can be tracked with instances of a special **ActivityRecord**, **ConfigurationEvent**.

Location class is not intended to be used for graphical data exchange, although its **PositionPoint**s can be used to derive initial layouts for graphical applications. Exchange of single line diagrams representing **PowerSystemResource**s is supported by model package **IEC61970::DiagramLayout** defined in base CIM in IEC 61970-301.

NOTE Support for complex geo-spatial data exchanges is out of the scope of this part of IEC 61968, since there are widely used and adopted standards (e.g. GML) for other domains that can be applied to electrical networks as well.

Figure 4 shows how **Location** class is used in relation with **Asset**s, **PowerSystemResource**s and **UsagePoint**s.

---

9 In this part of IEC 61968 (2nd edition), all the DCIM classes that were defined in the model package **IEC61968::WiresExt** of IEC 61968-11:2010 (1st edition), have been moved to base CIM, IEC 61970-301 (5th edition) and model package **IEC61970::Wires**, to ensure functional integrity of the overall CIM network model, for both transmission and distribution networks.

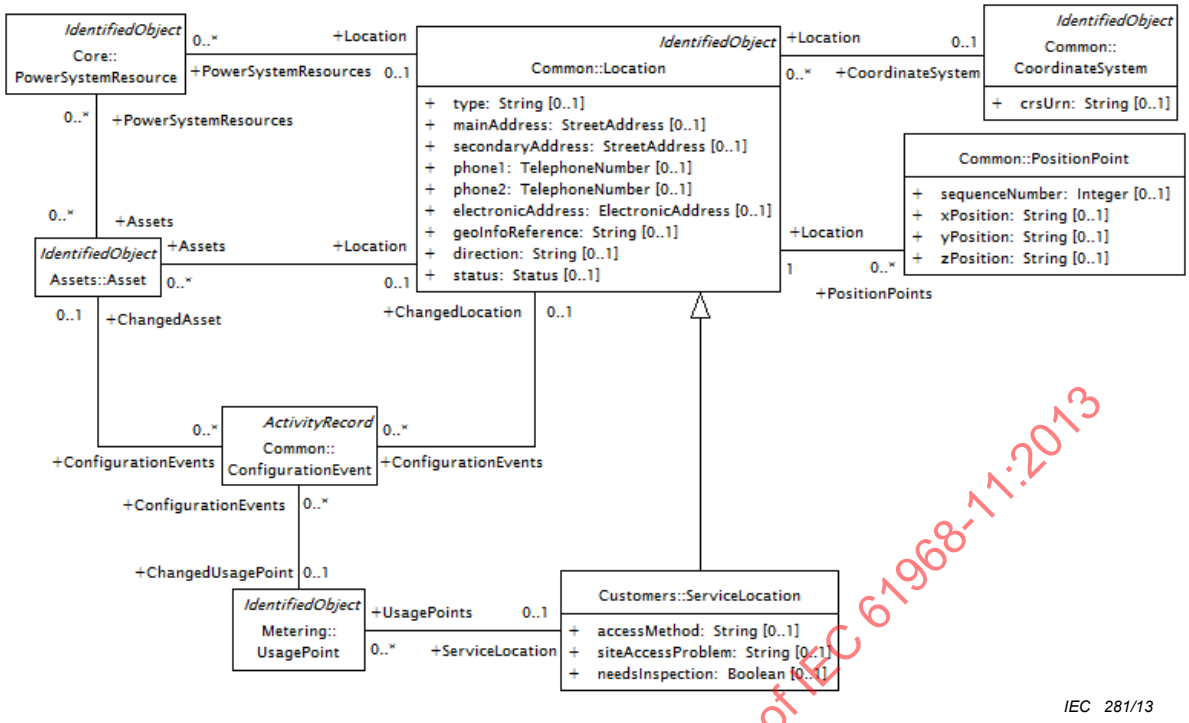


Figure 4 – DCIM power system resource and asset locations

An **Asset** or a **PowerSystemResource** directly associate with a **Location**, because the "general" **Location** provides all the information required (such as **mainAddress**, **electronicAddress**, or associated **PositionPoints**). In contrast, a **UsagePoint** references a specialised type of location, **ServiceLocation**, with additional attributes relevant for service only, such as **needsInspection**. This is a commonly used pattern in DCIM: To provide associations among specialisations where applicable and to relate high level classes where the generic relationship applies to all the specialisations.

4.4.2.3 Organisations

Figure 5 illustrates two major classes used to model organisations, **Organisation** and **OrganisationRole**, and their intended usage.

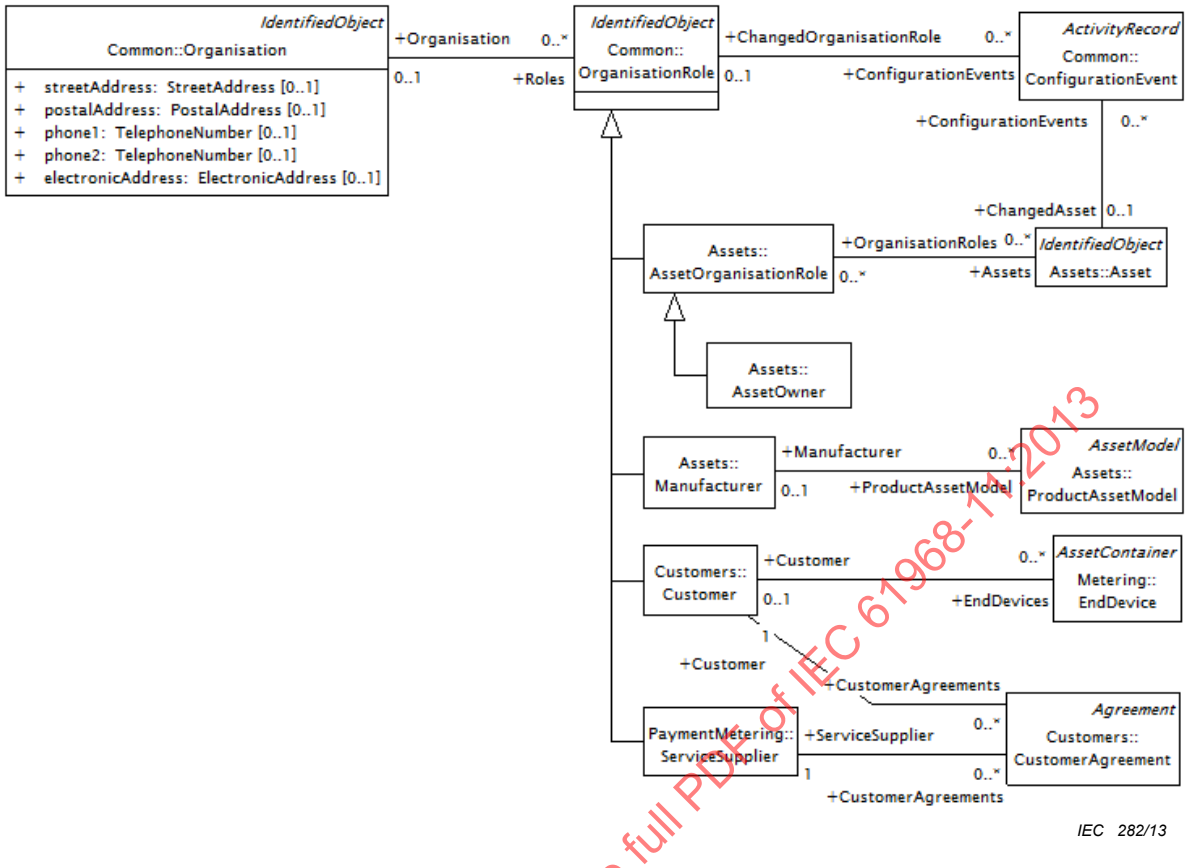


Figure 5 – DCIM organization model

Business entities modelled with DCIM are often used in many disparate contexts. For example, the same instance of **Organisation** may have the role of a **Customer** in a context dealing with service requests for **EndDevice**s, and the role of a **Manufacturer** of a **ProductAssetModel** in a context of asset maintenance; or the role of an **AssetOwner** in the context of asset management, and the role of a **ServiceSupplier** in the context of service requests and according to **CustomerAgreement**. This is modelled in DCIM with the association of an **Organisation** with multiple **OrganisationRole**s, whereas each **OrganisationRole** may reference a single **Organisation**, as a business entity.

Figure 5 shows the **OrganisationRole** as the common supertype of various roles. They are introduced on need, when there are specific attributes (not shown in the figure) and/or associations applicable to only certain domain of usage. One such example is **Manufacturer**: that role is applicable to a given **ProductAssetModel**, not to the **Asset** in general, and therefore there is an explicit association of that specific **OrganisationRole** specialisation with the **ProductAssetModel**, while there are other specific roles in relation to **Asset**s (e.g., **AssetOwner**).

In DCIM, **Organisation** as a business entity is never specialised.

Configuration changes of **OrganisationRole**s can also be tracked with instances of **ConfigurationEvent**.

4.4.2.4 Assets

4.4.2.4.1 Major classes in assets model

Whereas IEC 61968-11:2010 (1st edition) contained **Asset** and **AssetModel** classes required only in the context of metering (IEC 61968-9), this part of IEC 61968 (2nd edition) provides a more elaborate assets model, supporting the needs for data exchanges in the context of

network model exchanges (IEC 61968-4 and IEC 61968-13). Figure 6 gives an overview of major classes that are included in the DCIM assets model.

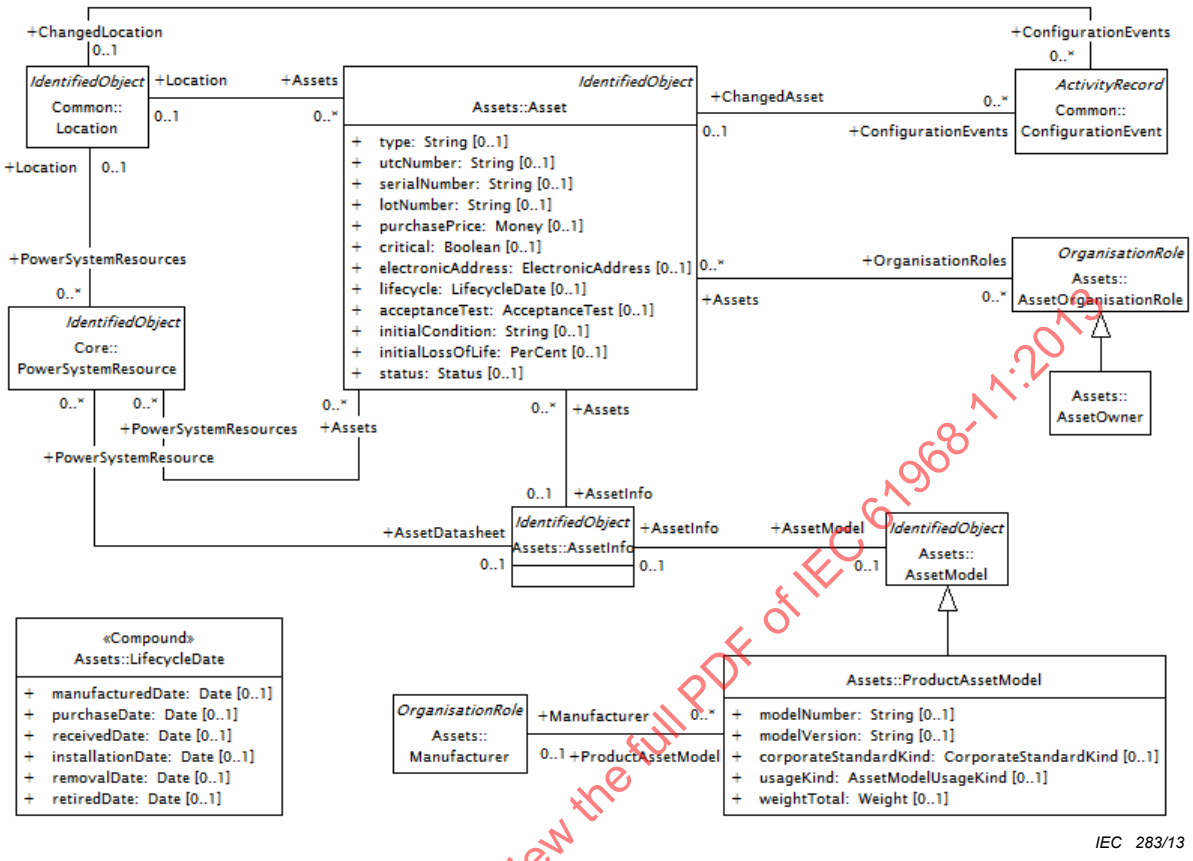


Figure 6 – DCIM assets model

Asset is a tangible resource of the utility, including power system equipment, various end devices, cabinets, vehicles, buildings, etc. An **Asset** instance is a managed resource that can be in operation or in stock, and that is characterised with its **lifecycle** attribute, the possible values being defined by the compound **LifecycleDate**. It may be described with a **Location** and its configuration may be tracked with instances of **ConfigurationEvent**. An Asset may be related to organisations through multiple **AssetOrganisationRole**s, such as **AssetOwner**.

NOTE In this part of IEC 61968 (2nd edition) (documenting DCIM11), there is a single specialisation of **AssetOrganisationRole**. It is anticipated that the next edition (documenting DCIM12) will add more organisation roles with respect to assets, such as maintainer or user, in support of data exchanges required by IEC 61968-3 and IEC 61968-6.

Some data exchanges will require electrical equipment connectivity model, in which case an **Asset** may reference one or more **PowerSystemResource** instances, which model the electrical function of the equipment represented by the physical **Asset** – see also 4.4.2.5. The inverse applies as well, namely, some data exchanges between operational and other systems (such as planning or asset management systems) will require a physical view associated with the electrical model of the power system, and in that case a **PowerSystemResource** may refer to the relevant **Asset** and/or **AssetInfo** instances. Finally, some data exchanges will completely ignore electrical connectivity model and will work exclusively with the assets model.

Another essential class in the assets model is **AssetInfo**, which is an identifiable container for various physical parameters describing associated **Asset**s. An **Asset** without the associated concrete **AssetInfo** instance is not fully described. **AssetInfo**, modelling what is sometimes called “catalogue”, “library” or “code”, is a set of attributes of an asset, representing typical

datasheet information of a physical device that can be instantiated and shared in different data exchange contexts:

- as attributes of an asset instance (installed or in stock);
- as attributes of an asset model (product by a manufacturer);
- as attributes of a type asset (generic type of an asset as used in designs/extension planning)[10].

Because datasheet attributes of physical devices depend on the type of device they describe, **AssetInfo** is the class that is specialised with concrete types that hold the attributes and associations specific to that specialisation (see 4.4.2.4.2).

The main characteristic of a concrete **AssetInfo** instance is the fact that it is possible to reference it from multiple instances of **PowerSystemResource** or **Asset**. This is a major requirement in distribution network modelling, where the equipment characteristics are rarely defined per **PowerSystemResource** or **Asset**, due to the nature and amount of the equipment used. Typically, one instance of a specific **AssetInfo** may be referenced by thousands of instances of **PowerSystemResource** or **Asset**. Similar requirement exists for calculated electrical parameters, such as line or transformer impedances, and this is reflected through the relevant electrical catalogue classes (see 4.4.3.3.1).

The third major abstraction related to assets is the concept of **AssetModel**. It represents the model of an **Asset**, either a product of a specific manufacturer (**ProductAssetModel**) or a generic asset model or material item (not shown in Figure 6, because that class is still informative). Datasheet characteristics of the **AssetModel** are available through the associated **AssetInfo** specialisation and can be shared with **Asset** or **PowerSystemResource** instances.

4.4.2.4.2 Specialisations in assets model

In the earlier versions of the electronic UML model of DCIM, there used to be three parallel inheritance hierarchies related to assets, in addition to the **PowerSystemResource** inheritance hierarchy of base CIM, in IEC 61970-301. This part of IEC 61968 (2nd edition) presents a simplified assets model, well consolidated with the function model of **PowerSystemResource**, and with the major classes introduced in 4.4.2.4.1.

4.4.2.4.2 gives the modelling guidelines that have been established and applied in this edition of IEC 61968-11, and that should be followed in the future standard editions of DCIM as well as for custom extensions of the standard DCIM.

a) Among classes **Asset**, **AssetModel**, **ProductAssetModel** and **AssetInfo**: **ProductAssetModel** should never be specialised in the standard DCIM, and **AssetModel** is specialised only with **ProductAssetModel**[11]. **AssetInfo** will always be specialised. **Asset** may be specialised according to guideline d) only.

b) If a device/equipment exists in **IEC61970::Wires** model package of IEC 61970-301 (as a specialisation of, or related to **PowerSystemResource**), it is not allowed to define its counterpart as a specialisation of **Asset**. In contrast, a specialisation of **AssetInfo** shall be defined with its datasheet properties, where applicable.

c) Relationship between functional and physical models is established between relevant specialisations of **PowerSystemResource** and **AssetInfo** through inherited relationship,

[10] Among informative classes in the electronic UML model of DCIM, there is one more specialisation of **AssetModel** which is anticipated to become normative in some of the later editions of IEC 61968-11.

[11] Among informative classes in the electronic UML model of DCIM, there is one more specialisation of **AssetModel** which is anticipated to become normative in some of the later editions of IEC 61968-11.

either **PowerSystemResource–AssetInfo** (if asset instance does not exist) or **PowerSystemResource–Asset–AssetInfo** (if asset instance is available).[12]

d) If there is a device/equipment/network-related entity not modelled in **IEC61970::Wires** model package of IEC 61970-301, it should be inheriting from **Asset** or **AssetContainer**, and its name need not contain the (now redundant) word "Asset".[13]

e) According to a), the specialisations of **Asset**, **AssetModel**, **ProductAssetModel** should be eliminated as follows: The required attributes and association ends should be moved into either an **Asset** specialisation (if they are applicable to an instance of **Asset**), or to an **AssetInfo** specialisation (if they represent datasheet information, applicable to many **Asset** instances).[14]

Figure 7 shows some DCIM examples that reflect the above guidelines and also illustrates the usage of some more classes from the assets model.

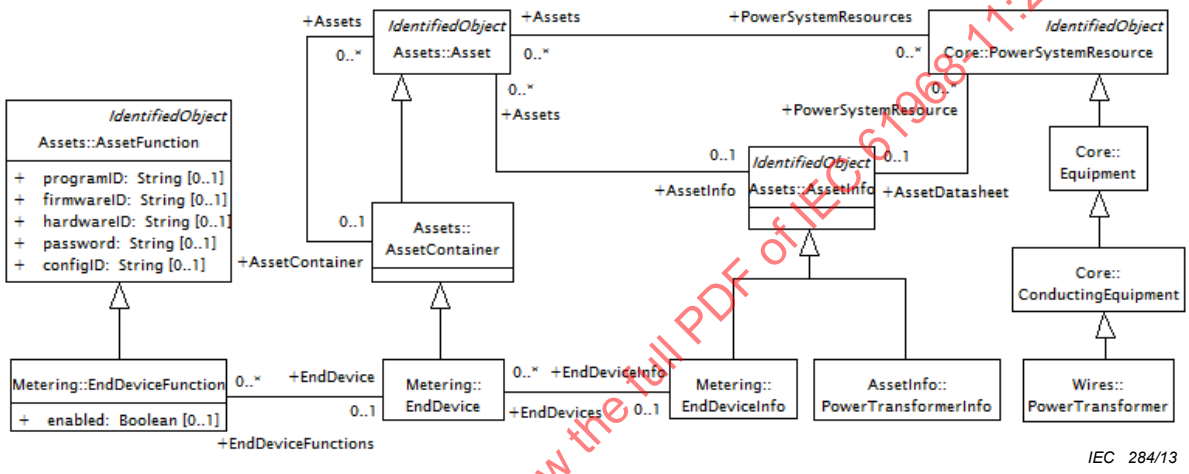


Figure 7 – DCIM assets – specialising guidelines

AssetContainer is the specialisation of **Asset**, and also the container for multiple **Asset**s. As a specialisation, it inherits the attributes and associations with other classes. As a container, it allows to model composite **Asset**s. One such example is the **EndDevice**, which may contain other entities that may have independent lifecycle (such as communication modules, not shown in the figure). Other examples of **AssetContainer**s would be towers or vaults (currently not in the normative DCIM). Neither of these real world objects is defined in the functional, **PowerSystemResource** model (guideline d)).

Figure 7 shows only two among many specialisations of **AssetInfo**: **EndDeviceInfo** and **PowerTransformerInfo**. The latter one inherits an association to **PowerTransformer**s either through **AssetInfo–PowerSystemResource** or **AssetInfo–Asset–PowerSystemResource** (guideline c); for details, see 4.4.3.4). There is no **Asset** specialisation corresponding to **PowerTransformer** (guideline b)).

For **EndDeviceInfo**, it is directly associated with multiple instances of **EndDevice**, and it also inherits the association **AssetInfo–Asset**. In such a case, it is the most specific relationship that should be used. At present, there are few specialisations of **Asset**, and it is acceptable to

[12] Due to application of this guideline, some direct associations between specialisations of **PowerSystemResource** and **AssetInfo** have been eliminated in favour of using the new inherited association defined between **PowerSystemResource** and **AssetInfo**.

[13] Due to application of this guideline, some classes from **Metering** model package have been renamed in order not to use the suffix "Asset".

[14] Due to application of this guideline, some classes from **Metering** model package have "lost" their instance attributes, which have been moved to the corresponding **AssetInfo** specialisation.

define such concrete associations among specialisation, even where the inherited association exists. For the case of relationships between **AssetInfo** and **PowerSystemResource** specialisations, they have been considered too numerous to define all the explicit relationships among them.

Finally, the class **AssetFunction** and its specialisation **EndDeviceFunction** demonstrate the preferred design, when feasible: to start with only direct associations between the relevant specialisations, such as **EndDevice**–**EndDeviceFunction** in Figure 7. In the future DCIM editions, if there are several cases where the need for relationship between the specialisations of **AssetFunction** and **Asset** or **AssetContainer** are identified, the concrete relationship would likely be replaced with a generic one.

4.4.2.5 Electrical model vs. physical model

The distribution CIM covers both electrical and physical representation of an object. The **PowerSystemResource** class models the electrical representation and is often used for network operation, monitoring, outage management, and operations planning, while the **Asset** class models an object's physical representation and is mainly used for asset and work management, but also for network extension planning, outage management and network studies. For instance, the physical representation may be essential in deriving attributes for the electrical representation, even if no explicit **Asset** class is used. For instance, 4.4.3.3.3 explains how the asset model can help calculate the electrical characteristic of a distribution line. The relationship between the two aspects of the electrical equipment also provides key information for extension planning, work management and outage management.

The relationship between **Asset** and **PowerSystemResource** allows for navigation between the two different representations (models) of a real world object. Connectivity and operational aspects (such as operational limits or energisation status) are always modelled through **PowerSystemResource** and its specialisations, while physical aspects (such as data sheet ratings, dimensions or asset lifecycle and financial data) are always modelled through **Asset** and its related classes.

It can be noticed that both **Asset** and **AssetInfo** have relationships to **PowerSystemResource**. The relationship between **PowerSystemResource** and **Asset** is typically used for data exchanges when the **Asset** is in operation, to give the physical view of electrical network equipment being operated, or in the other direction, to give the electrical connectivity view to the system working with physical representation. The relationship between **PowerSystemResource** and **AssetInfo** can be used even if there is no actual **Asset** instance available or needed; typical example are planning applications, or network calculation applications used in operational systems (such as distribution power flows).

4.4.3 Network modelling concepts and examples

4.4.3.1 Connectivity and phase modelling

The distribution CIM connectivity model is identical with the base CIM connectivity model, i.e., it relies on **ConductingEquipment**, **Terminal**, and **ConnectivityNode** classes (refer to IEC 61970-301:—, 4.4.4, Connectivity model and 4.4.8, Phase wire modelling). With the **Terminal.phases** attribute, phase information of each **ConductingEquipment** can be expressed for a multi-phase distribution network. It also allows one to represent normal network phase mismatch cases. In distribution networks, there are some rare cases where a normal open switch may be connected to the phase A on one side and the phase B on the other side. **Jumper**s could introduce normal phase mismatch as long as one side is de-energised before the **Jumper** is connected. For instance, a **Jumper** could connect an energised normal phase A to a normal phase B that is currently de-energised. This case is only allowed when all downstream loads are on the same phase(s).

For an extensive example of balanced three-phase sample network, refer to IEC 61970-301:—, 4.4.4.2, Connectivity and containment example, and for an example illustrating how the CIM

connectivity model can be used to represent a multi-phase network, refer to IEC 61970-301:—, 4.4.8, Phase wire modelling.

Figure 8 gives an overview of phase-related classes used typically for distribution network modelling and applications.

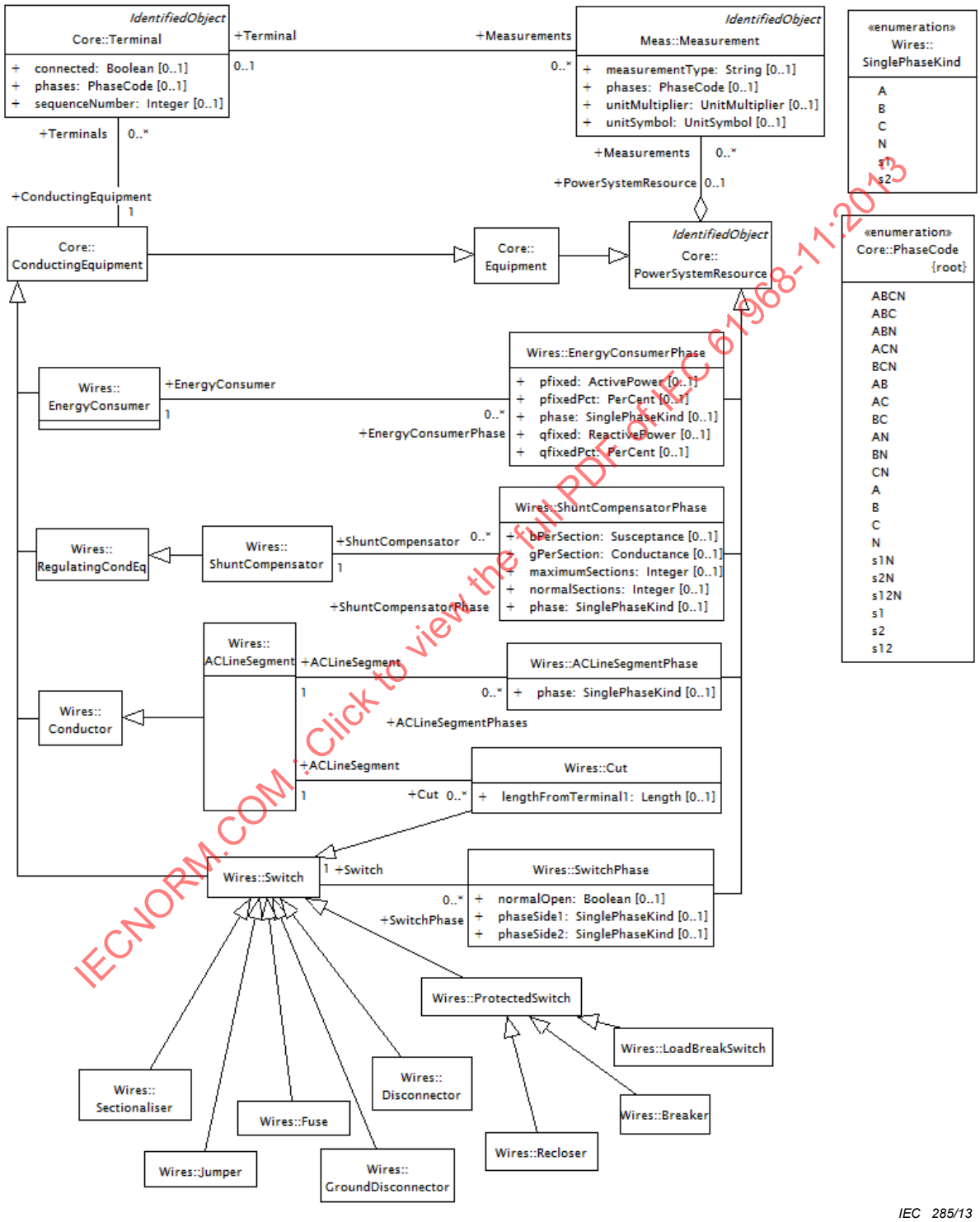


Figure 8 – DCIM phase modelling

In distribution networks, some of the devices could have different electrical and operating characteristics on each phase. For instance, a three-phase **Switch** (whether it is a **Breaker**, a

Fuse, or any other specialisation of **Switch**) may be composed of three separate physical switches, one for each phase. In the majority of cases the three individual switches have the same physical properties. If such a switch is gang operated (i.e., all the physical switches have the same normal and/or the real-time operational state), it can be modelled with a single instance of **Switch**. However, it may happen that the individual phases have different physical properties. For instance, in the case of a **Fuse**, at least theoretically, different fuse ratings might be used on each phase. Also, for un-ganged switches there may be cases where the normal and/or the real-time operational state of each phase could be different. This case is then modelled as one instance of **Switch** containing three instances of **SwitchPhase**, one for each phase. Important point is that there is one **Switch** instance in every case, and that one instance can be referenced from different contexts (e.g., configuration, operations).

NOTE This part of IEC 61968 does not support multi-state switches (modelled by **CompositeSwitch**).

From this part of IEC 61968, similar to **Terminal** class, **Measurement** class has defined the **phases** attribute to provide detailed phase description for a measurement. In most cases the attribute is optional since the measurement on a **Terminal** matches its phase. However, for a **ConductingEquipment** that could be operated separately on each individual phase, the **Measurement.phases** attribute provides the capability to specify measurements on each phase. Based on the CIM control model defined in base CIM, IEC 61970-301:—, 4.4.10, Measurements and controls, controls can be applied to separate phases as well.

From this part of of IEC 61968, the distribution CIM model provides support for representing temporary changes in the network. For detailed description and example, refer to IEC 61970-301:—, 4.4.9, Cuts, clamps and jumpers model.

4.4.3.2 Single-phase and unbalanced loads

Figure 9 shows the classes available to model distribution loads, which are often unbalanced among the three phases at a location. In some cases, single-phase and two-phase loads will occur.

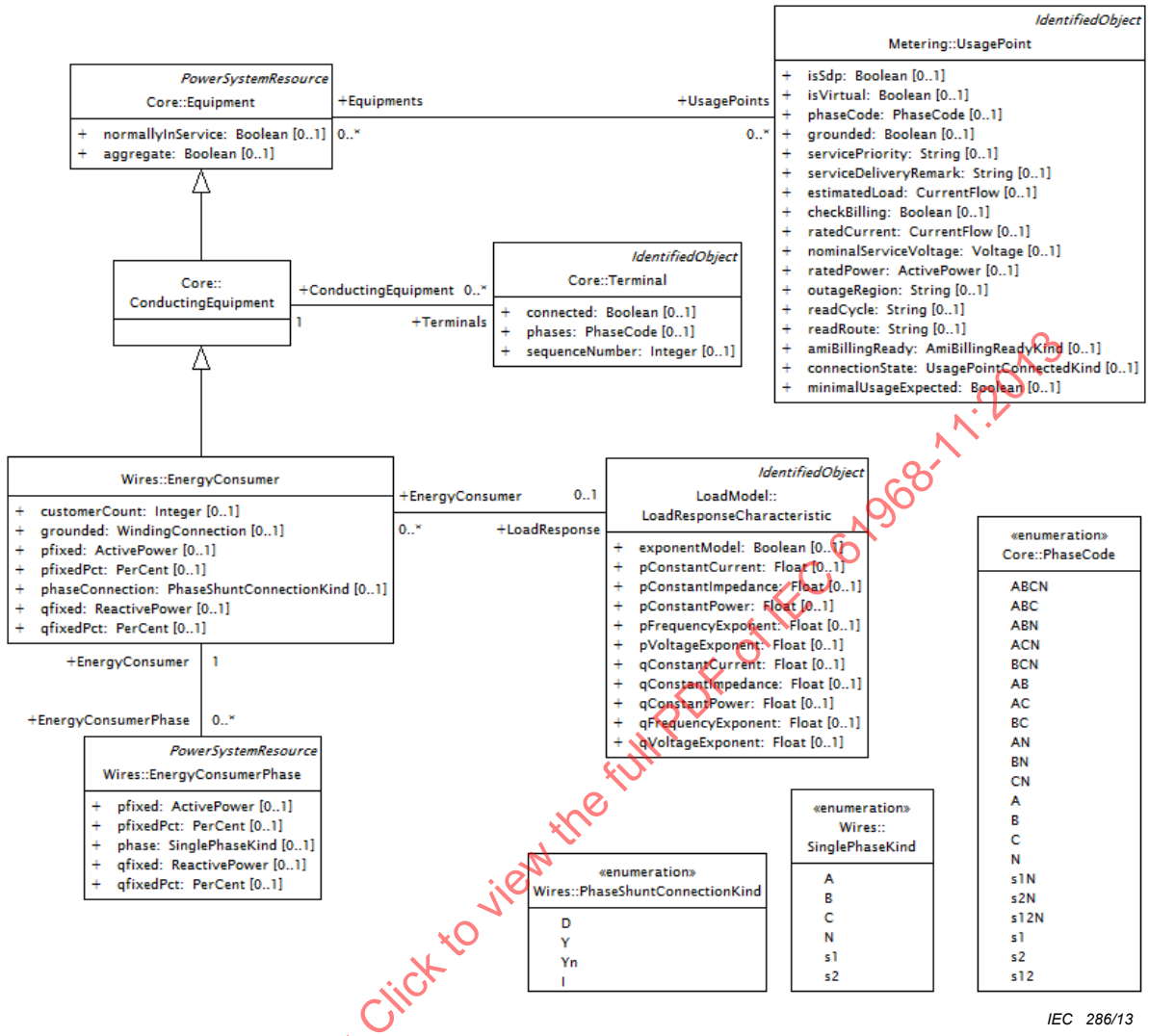


Figure 9 – DCIM load model

The **EnergyConsumer** class should be used to instantiate the load model, which can optionally be associated with a meter through **UsagePoint**. **EnergyConsumer** inherits from **ConductingEquipment** which is associated with **Terminal**. **Terminal** has **phases** attribute, which may be assigned a **PhaseCode** enumeration literal. For example, **AN** describes a single-phase load from A to neutral, **BC** describes a single-phase load from B to C, **ABCN** describes a three-phase, wye-grounded load, **ABC** describes a three-phase delta-connected load, etc.

The **phaseConnection** attribute of **EnergyConsumer** should be **Y** or **Yn** for a wye connected or phase-to-neutral load; it should be **D** for a delta or phase-to-phase load. For a balanced three-phase load, specify the total real and reactive power on **EnergyConsumer** attributes, for equal distribution among the three phases.

If a three-phase load is unbalanced, it can still be modelled with the same **EnergyConsumer** instance, now referring to three additional **EnergyConsumerPhase** instances, each representing one phase with its **phase** attribute. This makes it possible to use an **EnergyConsumer** with its identity for unbalanced modelling as well. With single-phase or two-phase loads, the model should also include correctly phased conductors or transformers that establish connectivity back to the source. The total real and reactive power need not be specified on **EnergyConsumer** attributes, because they can be derived from the distribution among phases specified on **EnergyConsumerPhase** attributes. Each **phase** can be **A**, **B**, **C**, **S1**,

or **S2** (**N** does not apply to this case). For a delta or phase-to-phase connected load, which is indicated with **D** for **phaseConnection**:

- use **A** for the load connected between phases **A** and **B**;
- use **B** for the load connected between phases **B** and **C**;
- use **C** for the load connected between phases **C** and **A**;
- use **S1** for the load connected between phases **S1** and **S2**.

In case the load is a combination of constant current, constant power, or constant impedance, an instance of **LoadResponseCharacteristic** should be associated with the **EnergyConsumer** instance.

NOTE This part of IEC 61968 does not provide the means to define different load characteristics per phase.

4.4.3.3 Distribution line segments

Figure 10 shows the classes available to model AC line segments (i.e. conductors). There are three ways to describe the impedance parameters of an **ACLineSegment** using only the **Wires** package, and a fourth way making use of the **AssetInfo** package.

NOTE This part of IEC 61968 (documenting DCIM11) and the corresponding IEC 61970-301 (documenting base CIM15) reflect consolidated line segment models for transmission and distribution (T&D). Therefore, the classes from IEC 61968-11:2010 (1st edition) have been slightly modified and moved from model package **IEC61968::WiresExt** into model package **IEC61970::Wires** of the base CIM, IEC 61970-301 (5th edition).

In order to represent a single-phasesingle-phase, two-phase, or unbalanced three-phase line using different wires, one **ACLineSegmentPhase** instance should be associated with each phase or neutral wire retained in the model. The applicable **phase** attribute values are **A**, **B**, **C** for three-phase lines, **S1** and **S2** for single-phase secondary circuits, and **N** for cases where the neutral wire is modelled. Each **ACLineSegmentPhase** has an optional association to **WireInfo**, described later in more detail, but all other characteristics come from its associated **ACLineSegment**.

For an unbalanced line, the calculated impedances reflect physical wire positions on the tower or pole, and these positions are typically ordered from left-to-right and top-to-bottom based on their horizontal and height coordinates. For example, consider a pole with crossarm having 3 wire positions numbered 1 (left-most), 2 (offset from centre), and 3 (right-most), impedances could be calculated with phase A's wire in position 1, phase B's in position 2, and phase C's in position 3. A different **ACLineSegment** might use the same pole type and wires, but with phase C's wire in position 1, phase B's in position 2, and phase A's in position 3. The calculated impedance parameters will be different than the first case. They are mathematically related by impedance matrix row and column operations, but such operations are not supported in the CIM, nor is there a concept of ordering phases in CIM. This means that each different physical phase ordering requires a different impedance description in the CIM. This consideration affects the use of **PerLengthPhaseImpedance** and **WireSpacingInfo** in Figure 10.

Transposed lines can be modelled with a different **ACLineSegment** for each section, and a different impedance description for each section as just discussed. The impedance description should use either **PerLengthPhaseImpedance** or **WireSpacingInfo**, because sequence parameters assume continuous transposition. It is not necessary to use **ACLineSegmentPhase** instances for transposed three-phase line sections, if all phase wire types are identical within a section.

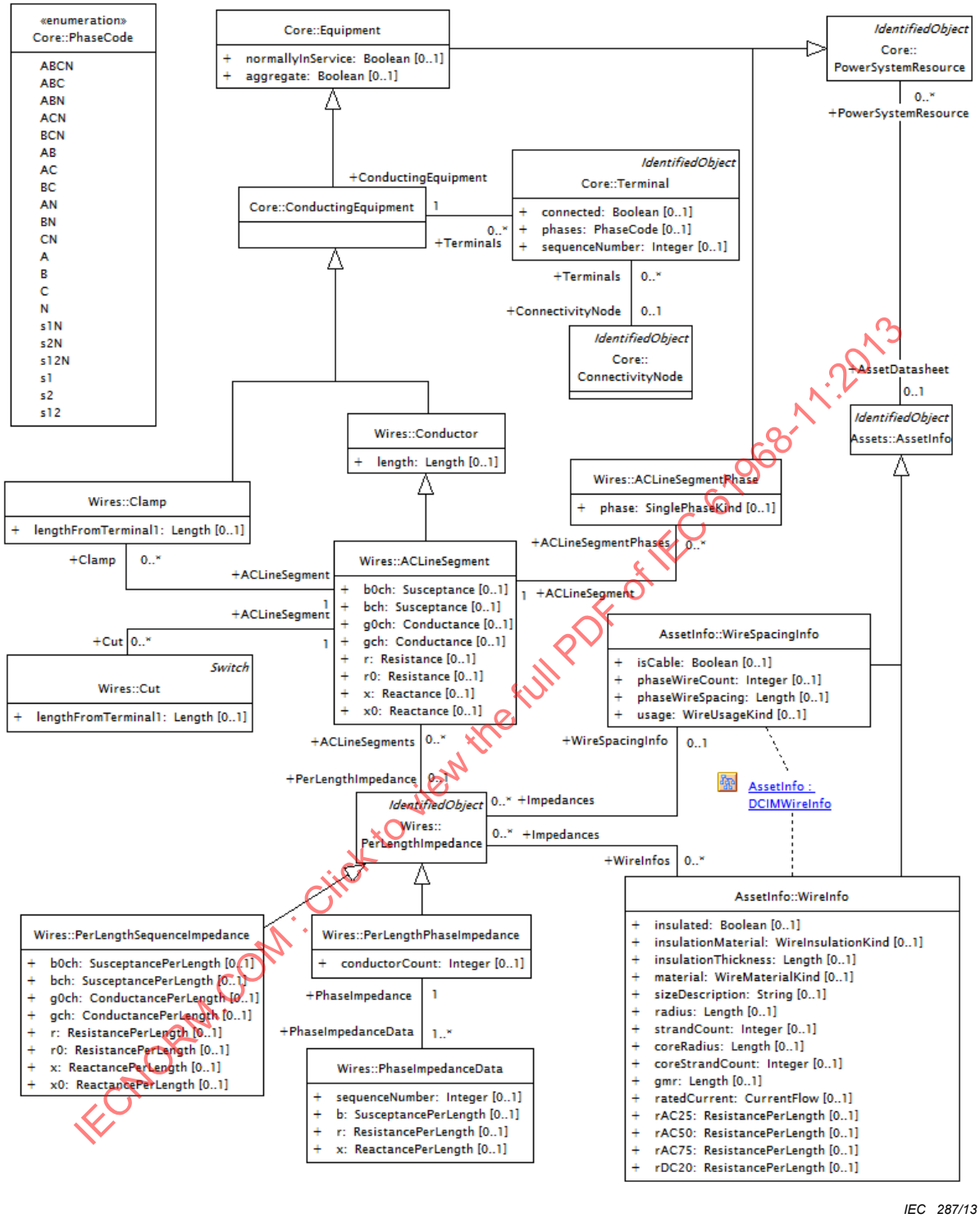


Figure 10 – DCIM line connectivity model

There are three ways to specify line impedances using only the **IEC61970::Wires** model package from IEC 61970-301:

- The **ACLineSegment** attributes **r**, **r0**, **x**, **x0**, **bch**, **bch0**, **gch**, and **gch0** may be used for a balanced model. See 4.4.3.3.1 for usage with single-phase and two-phase line segments.
- Provide an association to **PerLengthSequenceImpedance**. This class models an electrical catalogue, or a library of "line codes" that have sequence impedances and line charging per unit length. Multiple **ACLineSegments** will share one instance of **PerLengthSequenceImpedance** and will calculate impedance with their **length** attribute

inherited from **Conductor**. See 4.4.3.3.1 for usage of sequence parameters with single-phase and two-phase line segments.

- Provide an association to **PerLengthPhaseImpedance**, which references pre-calculated NxN symmetric impedance and admittance matrices per unit length. This class also models an electrical catalogue and the inherited **length** attribute is required. The **conductorCount** has to be at least equal to the number of phases, but it could be higher if the neutral (or other grounded conductor) is retained in the matrix. **PhaseImpedanceData** implements Z and Y matrix elements, stored in column order. The attributes **r**, **x**, and **sequenceNumber** are all required, while **b** is optional. Only the lower triangular elements are stored, so that a 3x3 matrix would have 6 elements (i.e., instances of **PhaseImpedanceData**). The matrix rows and columns have to be in phase order. A referencing **ACLineSegment** will assign row 1 to the first phase present from the ordered list (**A**, **B**, **C**, **s1**, **s2**, **N**), row 2 to the next phase present, and so on. See also 4.4.3.3.2 below.

To specify impedances using the **AssetInfo** package, the **Conductor.length** attribute is required, because the instance electrical parameters are defined as (per-unit length parameters) × (**Conductor.length**). See also 4.4.3.3.3 below.

A **WireSpacingInfo** can be associated to the **ACLineSegment** using either of two methods:

- (**PowerSystemResource**–**AssetInfo**) Use the inherited **AssetDatasheet** association end of **ACLineSegment** to reference **WireSpacingInfo**. This method can also be used to associate **WireInfo** for its **ratedCurrent** attribute.
- (**PowerSystemResource**–**Asset**–**AssetInfo**) Instantiate the **Asset** class, in which **AssetInfo** association end references the **WireSpacingInfo**, and multiple **PowerSystemResource**s reference each **ACLineSegment** using that **WireSpacingInfo**.

For impedance calculations, it is also necessary to associate **WireInfo**s to each referencing **ACLineSegment** or **ACLineSegmentPhase**. Either **AssetDatasheet** (**PowerSystemResource**–**AssetInfo**) or **Asset** class (**PowerSystemResource**–**Asset**–**AssetInfo**) navigation pattern accomplishes this. Whenever the **ACLineSegment** has associated **ACLineSegmentPhases**, each **ACLineSegmentPhase** should have its own **WireInfo**. If not:

- At least one **WireInfo** shall be associated to the **ACLineSegment**, and it is assumed to apply for all phase wires. It also applies to any neutral wires, unless a second, identifiable **WireInfo** can be associated.
- An optional second **WireInfo** may be associated for the neutral wire, identified as the one with smallest **WireInfo.radius** attribute value. Whenever this assumption does not apply to the line, use by-phase modelling with **ACLineSegmentPhase**.

Figure 10 shows associations between **PerLengthImpedance**, **WireSpacingInfo**, and **WireInfo**. However, these are for asset library management and not for impedance calculations. Either the **AssetDatasheet** or **Asset** class pattern is required to use **WireSpacingInfo** and **WireInfo** for impedance calculations.

Figure 11 shows the **WireInfo** class hierarchy, used for defining line segment impedances from physical data (sometimes referred to as “line codes” and “cable codes”). **WireInfo** is a base class that should not be instantiated directly. Its attributes describe the physical data for an overhead wire (note that derived **OverheadWireInfo** currently has no attributes of its own), or a cable’s core phase conductor.

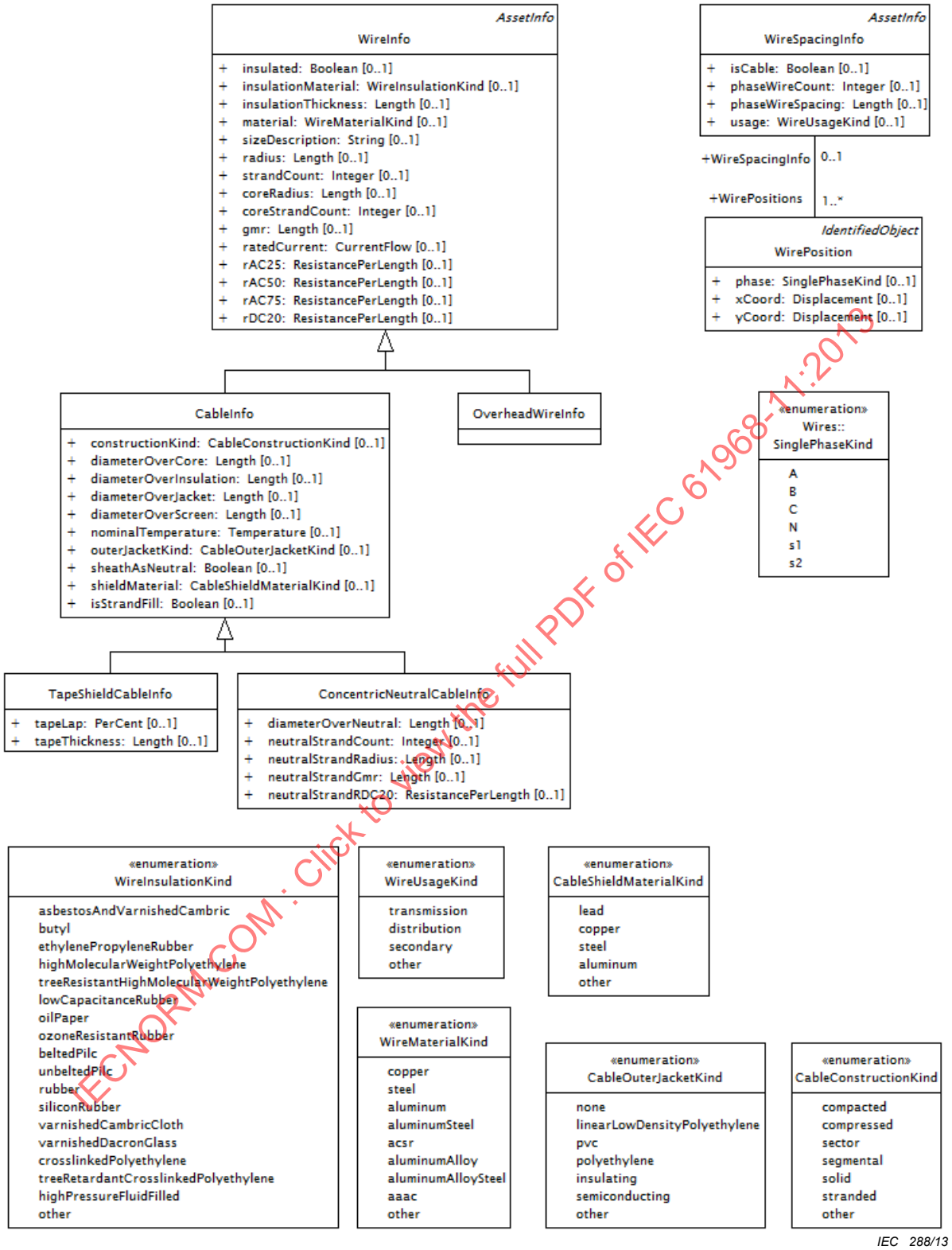


Figure 11 – DCIM conductor (line and cable datasheet) model

WireInfo attributes **material**, **sizeDescription**, **strandCount**, and **coreStrandCount** (for ACSR, aluminium conductor steel reinforced wire) help to identify the wire, and are often coded into the instance local name. The minimum required electrical attributes are **gmr**, **radius**, and **rAC50**. A complete wire table would usually have resistances defined at other temperatures and frequencies, such as **rAC25**, **rAC75**, and **rDC20**. For ACSR wires, the **coreRadius** is optional for frequency-dependent calculation of the **gmr.** The **coreRadius** is zero for non-ACSR wires. The **ratedCurrent** is the ampacity at 50 ˚C. The **ratedCurrent** is necessary to interpret

power flow output, but not for the power flow solution itself. The **insulated**, **insulationMaterial**, and **insulationThickness** attributes apply mainly to the derived cable classes, but they also support triplex secondary lines and overhead spacer cable. The **CableInfo** attributes describe additional properties of layers over the conducting core. The **diameterOverCore** includes both conductor and semi-conducting screen; it should be the insulation's inside diameter. The **diameterOverInsulation** is the insulating layer's outside diameter, not including any outside screens or semi-conducting layers. The **diameterOverScreen** includes the insulation plus screen or semi-conducting layer; it should be the shield's or sheath's inside diameter. The **diameterOverJacket** is the cable's outside diameter; it should be the largest diameter value specified except for concentric neutrals.

The **TapeShieldCableInfo** attributes **tapeLap** and **tapeThickness** describe a thin tape covering the cable, usually made of copper, in overlapping turns. The **ConcentricNeutralCableInfo** has attributes for the outer neutral conductors, which typically consist of several solid copper wires. The number of wires is **neutralStrandCount**, the radius is **neutralStrandRadius**, the geometric mean radius is **neutralStrandGmr**, and the resistance per wire is **neutralStrandRDC20**. These wires are so small that AC resistance is typically not used or specified. The **diameterOverNeutral** is the diameter over the concentric neutral strands, which may be the same as **diameterOverJacket**.

NOTE The model shown in Figure 11 currently supports the two types of single-conductor cable most commonly found on distribution systems. Many other cable types found in transmission or industrial systems are not supported; these include 3-conductor, pipe-type, submarine, and others.

WireSpacingInfo identifies the line geometry data. The **phaseWireCount** and **phaseWireSpacing** attributes refer to sub-conductor bundling, which is not common for distribution lines, but may appear on high-voltage transmission lines in the model.

WirePosition defines the horizontal (**xCoord**) and vertical (**yCoord**) coordinates of each wire on the pole or tower cross section, or in the cable trench/duct, and **phase** identifies which phase wire is mounted there. All three attributes are required. The overhead wire height above ground is **yCoord**, including any sag effects. For underground cables, **yCoord** is the average burial depth, entered as a negative number. The wire horizontal position, **xCoord**, is measured from an arbitrary but consistent reference line. Common choices for the horizontal reference are the pole centreline, and the left-most wire position.

Any wires at a **WirePosition** with **phase=N** are assumed to be continuously grounded. The application may eliminate these conductors from the impedance and admittance matrices through Kron reduction.

Many times, the **WirePosition.phase** attribute value could change depending on the field installation. For example, a single-phase line can be used on phase **A**, **B**, or **C**. This requires three **WireSpacingInfo** instances, the only difference being three different values for one of the associated **WirePosition.phase** values, namely **A**, **B**, or **C**. The **WirePostion.phase** value for a neutral wire would not change; it should always be **N**. In summary, a single-phase line type requires 3 **WireSpacingInfo** instances to cover all phasing possibilities. A two-phase line type requires 6 instances, and a three-phase line type also requires 6 instances.

4.4.3.3.1 Using sequence impedances (balanced case)

The positive and zero sequence impedances may be transferred through the **r**, **x**, **r0**, and **x0** attributes of **PerLengthSequenceImpedance** associated with an **ACLineSegment** instance. The **bch**, **b0ch**, **gch**, and **g0ch** attributes are usually not important for overhead distribution lines. For three phases, this describes a balanced three-phase, or perfectly transposed, line. The attributes in **PerLengthSequenceImpedance** are expressed in units per length, so it is necessary to multiply their values with the **length** attribute of **Conductor**.

NOTE This is equivalent to the attributes of **Wires::ACLineSegment**, which are pre-calculated for the whole length of the segment and are defined on each instance of the segment. In contrast,

PerLengthSequenceImpedance is referenceable, and as such can be used (through association) by several segment instances, thus decreasing the amount of data transferred in data exchanges.

If the **ACLineSegment** has only one or two phases, a balanced model can still be transferred through the **r**, **x**, **r0**, and **x0** attributes. This represents an impedance matrix with equal complex diagonal elements, Z_s, and equal complex off-diagonal elements, Z_m. For a single-phase line, the attributes to transfer are:

$$Z_1 = Z_0 = Z_s$$

For a two-phase or three-phase line, the attributes to transfer are:

$$Z_1 = Z_s - Z_m$$

$$Z_0 = Z_s + (n - 1)\, Z_m$$

where n is the number of phases. Upon receipt of **r**, **x**, **r0**, and **x0**, the balanced two-phase or three-phase impedance matrix is constructed from:

$$Z_s = (Z_0 + (n - 1)\, Z_1) / n$$

$$Z_m = (Z_0 - Z_1) / n$$

The referencing **ACLineSegment** should have associated instances of **ACLineSegmentPhase**, assigned appropriate **phase** values to show the phases actually present, such as **A**, **B**, **C**, **s1**, or **s2**. The neutral, **N**, should not appear because any neutral conductor must have been incorporated into the earth return when sequence impedances are used.

For underground distribution cables, the sequence impedances are also appropriate, including the **bch** and **b0ch** attributes.

4.4.3.3.2 Using phase impedances (unbalanced case)

Calculated matrix parameters may be transferred by referencing a **PerLengthPhaseImpedance** instance, in lieu of the physical model described in 4.4.3.3.3. A matrix model is useful when:

- the target application has no means of calculating parameters from the physical data;
- the underlying physical data is not readily available;
- it is necessary to match the unbalanced line parameters as closely as possible.

For a two-phase line, with the neutral already reduced, the Z and Y matrices will be 2x2, but due to symmetry, there will only be 3 unique matrix elements. That leads to three associated instances of **PhaseImpedanceData** with column-wise storage:

- **sequenceNumber** = 1 for row 1, column 1 of the matrix;
- **sequenceNumber** = 2 for row 2, column 1 of the matrix;
- **sequenceNumber** = 3 for row 2, column 2 of the matrix.

This instance of **PerLengthPhaseImpedance** could be referenced from an **ACLineSegment** having phases **AB**, **AC**, or **BC**. Row 1 always corresponds to the first phase present and row 2 always corresponds to the second phase present, following the order: **A**, **B**, **C**, **s1**, **s2**, **N**. If the phase wires are installed in different positions, such that **C** (for example) should be in row 1, a new instance of **PerLengthPhaseImpedance** is required.

4.4.3.3.3 Using physical parameters

For overhead lines, a physical model can be transferred through reference to a **WireSpacingInfo** instance, which is associated with further classes shown in Figure 11. This will support calculation of an unbalanced phase impedance matrix through the use of Carson's equations, or an equivalent method of handling the earth return. For example, suppose there are three phase wires, plus a different size neutral wire, on a pole with horizontal crossarm. This requires one **WireSpacingInfo** instance, four **WirePosition** instances that describe the four conductor positions, and two **WireInfo** instances describing the phase and neutral wire types. The **length** attribute of **Conductor** shall be used, and many **ACLineSegment**s will typically refer to the same **WireSpacingInfo** instance.

The **WirePosition** instances have a **phase** attribute that maps to **ACLineSegmentPhase.phase**. The resistance attribute of **WireInfo** should be supplied for fundamental power frequency, and at the wire's desired operating temperature for calculations.

A single-phase, concentric neutral cable requires one instance of **ConcentricNeutralCableInfo**, one **WireSpacingInfo** with attribute **isCable**=true, and one **WirePosition** instance to specify the burial depth. A three-phase tape shielded cable, with bare neutral conductor, would require one instance of **TapeShieldCableInfo**, one **WireSpacingInfo** with attribute **isCable**=true and four **WirePosition**s for the three phases and neutral. There would also be one **WireInfo** for the neutral.

4.4.3.4 Distribution transformers

4.4.3.4.1 Electrical model

Figure 12 shows the classes that model power transformer instances. They can use the **Wires** package exclusively to define impedance parameters, or make use of the **AssetInfo** classes detailed in Figure 13, to define a library of transformer types.

NOTE This part of IEC 61968 (documenting DCIM11) and the corresponding IEC 61970-301 (documenting base CIM15) reflect consolidated transformer models for transmission and distribution (T&D). Therefore, the classes from IEC 61968-11:2010 (1st edition) have been slightly modified and moved from model package **IEC61968::WiresExt** into model package **IEC61970::Wires** of the base CIM, IEC 61970-301 (5th edition).

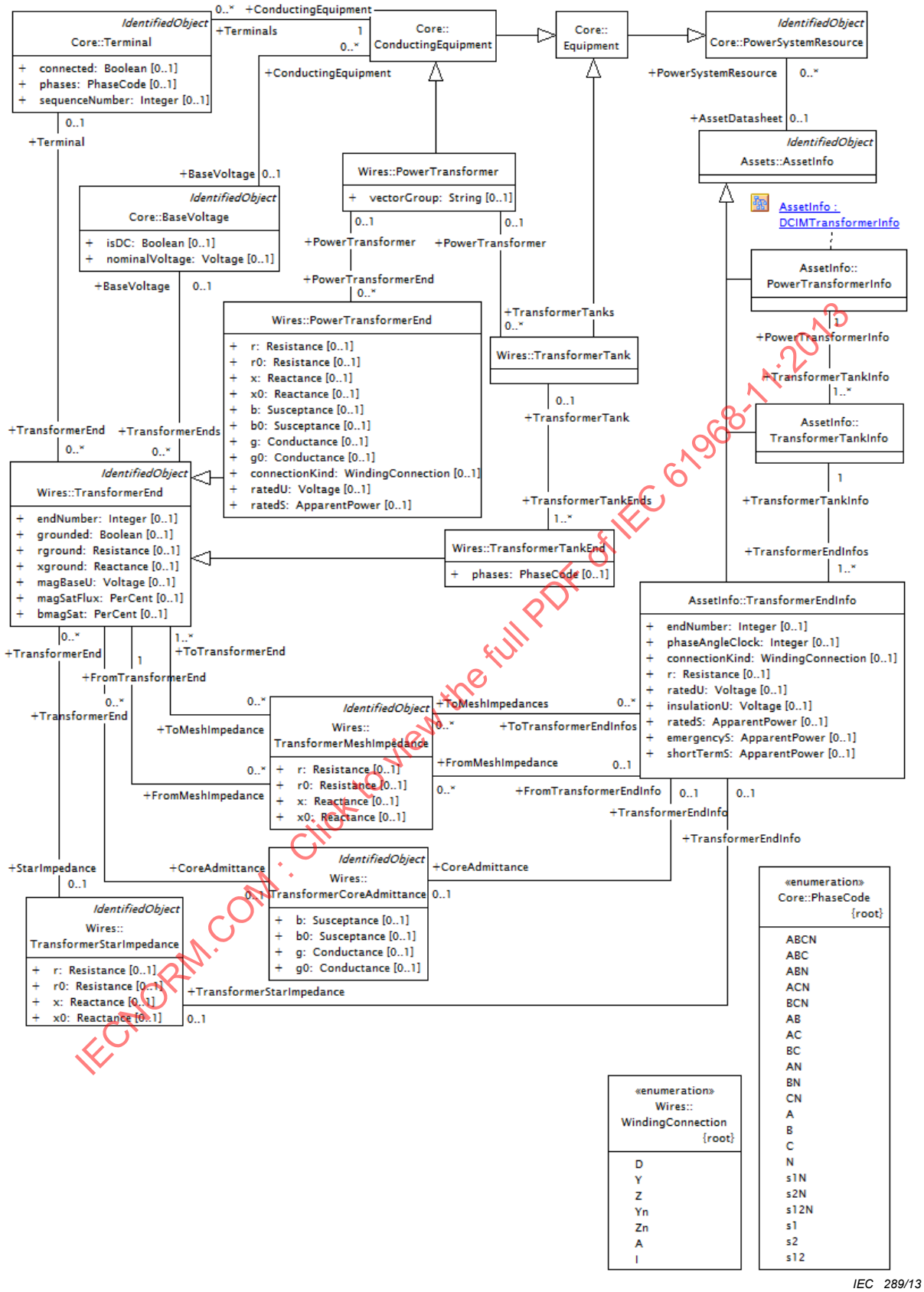


Figure 12 – DCIM transformer connectivity model

PowerTransformer is the top-most instance for a power transformer, whether composed of a three-phase tank, or possibly different single-phase tanks. It descends from

ConductingEquipment, and therefore, has associated **Terminal**s with **phases** attribute values at each winding connection point. In the CIM, a transformer winding is referred to as an “end”.

When composed of different tanks, a **PowerTransformer** has often been called a “transformer bank”, and the CIM supports modelling with or without tanks. In distribution systems, independent phase voltage regulators and open wye/open delta transformers provide two examples that require tank-level modelling. At the transmission level, EHV transformer banks may also contain single-phase transformers, which need not be identical, especially when a spare is in service. The **vectorGroup** attribute for protective relaying is derived from the internal winding connections and phase angles; it uses IEC 60076-1 nomenclature to describe any number of windings that may be included in the bank.

When used, the **TransformerTank** must be associated with a **PowerTransformer**. It inherits from **Equipment**, not **ConductingEquipment**. The tank may have associated **TransformerTankInfo** from the **AssetInfo** package, for asset datasheet modelling, described in more detail later with Figure 13 and 4.4.3.4.2. Because transformer testing is done on tanks, the datasheets are fundamentally associated with tanks. When not using tanks in the model, the **PowerTransformer** can still have an association to **PowerTransformerInfo**, for asset datasheet modelling. In both cases, the data actually resides in **TransformerEndInfo** instances, associated to a **TransformerTankInfo**.

There are two methods of referencing transformer asset datasheets, and a profile may prefer one over the other:

- (**PowerSystemResource–AssetInfo**) Use the **AssetDatasheet** association end of either **PowerTransfomer** or **TransformerTank**, to reference either **PowerTransformerInfo** or **TransformerTankInfo**, respectively, or
- (**PowerSystemResource–Asset–AssetInfo**) Instantiate the **Asset** class, in which **AssetInfo** association end references either the **PowerTransformerInfo** or **TransfomerTankInfo**, and multiple **PowerSystemResource**s reference each **PowerTransformer** or **TransformerTank** using that datasheet.

TransformerEnd, which was called TransformerWinding in earlier versions of CIM and was a **ConductingEquipment**, is not anymore a **ConductingEquipment**, but it does have one associated **Terminal** with phasing information. The **magBaseU**, **magSatFlux**, and **bmagSat** attributes represent core saturation, typically modelled at no more than one of the ends. The other instance attributes define the grounding options:

- solidly grounded: **grounded** = true, **rground** = 0, **xground** = 0;
- impedance grounded: **grounded** = true, **rground** ≥ 0, **xground** ≥ 0;
- ungrounded: **grounded** = false.

TransformerEnd should not be instantiated directly; one of its two descendants should be instantiated:

- **PowerTransformerEnd**, if not using tank-level modelling. Specify the **ratedU** and **ratedS** attribute values for winding rating data, and **connectionKind** for the wye, delta, or other type of connection.
- **TransformerTankEnd**, if using tank-level modelling. The winding connection and rating values come from a **TransformerTankEndInfo** instance, which means that the **AssetInfo** package is required for tank-level modelling. The **phases** attribute supports by-phase tank modelling.

With **PowerTransformer** and **PowerTransformerEnd**, there are three ways of specifying impedance parameters using only the **IEC61970::Wires** model package from IEC 61970-301:

- Use the **r**, **x**, **r0**, **x0**, **b**, **b0**, **g**, and **g0** attributes of **PowerTransformerEnd** to specify pi impedance parameters. This is the option most compatible with earlier versions of IEC 61970-301, and there are some important conventions described in that standard.
- Use one associated **TransformerStarImpedance** for each **PowerTransformerEnd**, comprising a star equivalent (also sometimes called tee or wye equivalent). This can be mathematically exact for up to three ends (windings). However, negative attribute values may occur in the case of three ends. Optionally, use one **TransformerCoreAdmittance** associated to one of the **PowerTransformerEnd**s, representing the exciting current and core losses. This can be the lowest-voltage winding (i.e. closest to the core), or the winding that was actually subjected to a no-load test, if known. The reference voltage for all attribute values, which have units of ohms or siemens, must be **ratedU** for the end to which the impedance or admittance is connected.
- Use a **TransformerMeshImpedance** associated to each combination of **PowerTransformerEnd** pairs. There shall be (numberEnds-1) × numberEnds / 2 of these; for example, one **TransformerMeshImpedance** between two ends, three of them between three ends, six of them between four ends, etc. The advantages of a mesh model are: (a) it is mathematically exact for more than three ends, (b) it has no negative attribute values, and (c) it corresponds more directly with transformer short-circuit test data. The reference voltage shall be **ratedU** of the **FromTransformerEnd** (note the other end nearly always has a different rated voltage). Optionally, use one **TransformerCoreAdmittance** as described for the star equivalent.

4.4.3.4.2 Datasheet model

Figure 13 shows the classes that allow for exchange of transformer datasheet models, sometimes referred to as “transformer codes” in applications.

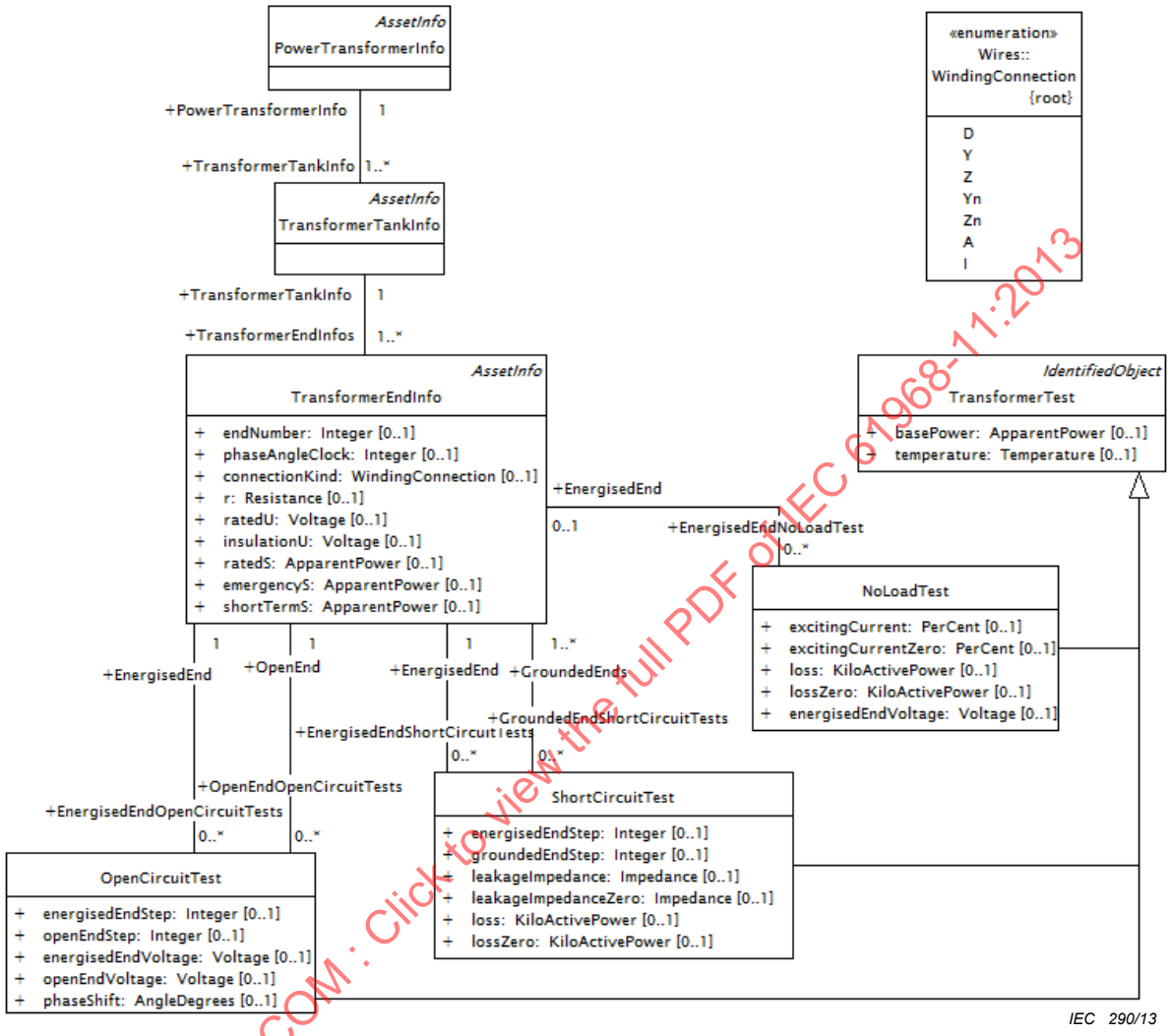


Figure 13 – DCIM transformer datasheet model

The main class is **TransformerEndInfo**, which contains rating and connection data for the corresponding transformer winding. Rating data includes **ratedU**, **ratedS**, **shortTermS**, **emergencyS**, and **insulationU**. The **r** attribute is the winding’s DC resistance.

Connection data is contained in attributes **connectionKind**, **phaseAngleClock** and **endNumber**. The **endNumber** attribute is the winding’s order in the **vectorGroup** of the referencing **PowerTransformer** and is used to map the data to **TransformerEnd.endNumber**. The associated **Terminal.sequenceNumber** could also use the same **TransformerEnd.endNumber**, but this is not required. By convention, the **endNumber** usually starts at 1 for the highest voltage winding, and all other windings are numbered in order of decreasing voltage rating. Some transformers have more than one winding with the same **ratedU**, split-secondary transformers providing just one example, and they shall have different **endNumber**s.

WindingConnection enumeration, used as type for the **connectionKind** attribute, includes the standard **D**, **Y**, **Z**, **Yn**, and **Zn** nomenclature to describe delta, wye, zigzag, and neutral connections in three-phase transformer vector groups. **A** is used for a common autotransformer winding, and **I** for a single-phase transformer winding.

The **endNumber** attribute allows the datasheet library to be used and updated with just one association to **PowerTransformerInfo** or **TransformerTankInfo**. The **TransformerTankInfo** and **PowerTransformerInfo** provide two navigation paths to **TransformerEndInfo**. With **PowerTransformerInfo**, it may be necessary to create an artificial **TransformerTankInfo** for model transfer, because datasheets are at the tank level. A **PowerTransformerInfo** instance actually refers to a collection of one or more tanks.

TransformerTankInfo is referenced by the **TransformerTank** (inherited **PowerSystemResource–AssetInfo**) and **PowerTransformerInfo** (directly). It serves mainly to organize the data into a library with one entry point. **TransformerTankInfo** collects associations to **TransformerEndInfo**, which generalizes to any number of windings.

There are two ways of specifying impedance parameters in a datasheet model:

- Associate **TransformerEndInfo** to the **TransformerMeshImpedance**, **TransformerStarImpedance**, and **TransformerCoreAdmittance** classes of the **IEC61970::Wires** model package from IEC 61970-301. These associations were shown in Figure 12 and described in 4.4.3.4.1.
- Use **TransformerTest** and its three descendant classes, shown in Figure 13. Each application is responsible for converting the test data to mesh equivalent, star equivalent, or some other electrical impedance model.

TransformerTest is the parent class for all of the transformer test classes, with **basePower** and **temperature** attributes applicable to all tests. **basePower** is essential for converting datasheet values to impedance or admittance at the correct reference voltage.

NoLoadTest's attributes **excitingCurrent**, **excitingCurrentZero**, **loss**, and **lossZero** are all measured on the **EnergisedEnd** winding and provide the basic data for a core admittance branch. Positive and zero sequence test data can be reported in the same instance of **NoLoadTest**. This test is generally done with rated voltage applied to the energised winding, but different values can be specified in **energisedEndVoltage**, and several tests may be provided to define core saturation parameters.

ShortCircuitTest is done by circulating rated current through the **EnergisedEnd** winding, with one or more **GroundedEnds** windings short circuited. It provides the basic data for the mesh or star equivalent circuit. If tests were done at different tap settings, the tap values are specified in **energisedEndStep** and **groundedEndStep**. Positive and zero sequence test data can be reported in the same instance of **ShortCircuitTest**. The AC resistances derived from **loss** and **lossZero** are likely to differ from the winding DC resistances, **TransformerEndInfo.r**, which are obtained from a separate test.

OpenCircuitTest's attributes **openEndVoltage** and **phaseShift** are measured on a single open winding, when the **EnergisedEnd** has **energisedEndVoltage** applied to it. Tap settings for both windings are specified in **energisedEndStep** and **openEndStep**. The tests are done to verify the transformer turns ratio, winding connections, and winding polarity. They are usually not required to determine electrical impedance parameters.

4.4.3.4.3 Tap changer model

Figure 14 shows the classes used to model a possibly unbalanced distribution voltage regulator. A **RatioTapChanger** is associated to a **TransformerEnd**. Each regulator uses autonomous local control, so that **RegulationSchedule** and **TapSchedule** (present in IEC 61970-301 and typically used in transmission) are not used here. Phase angle regulators and variation curves are also not generally used on distribution systems.

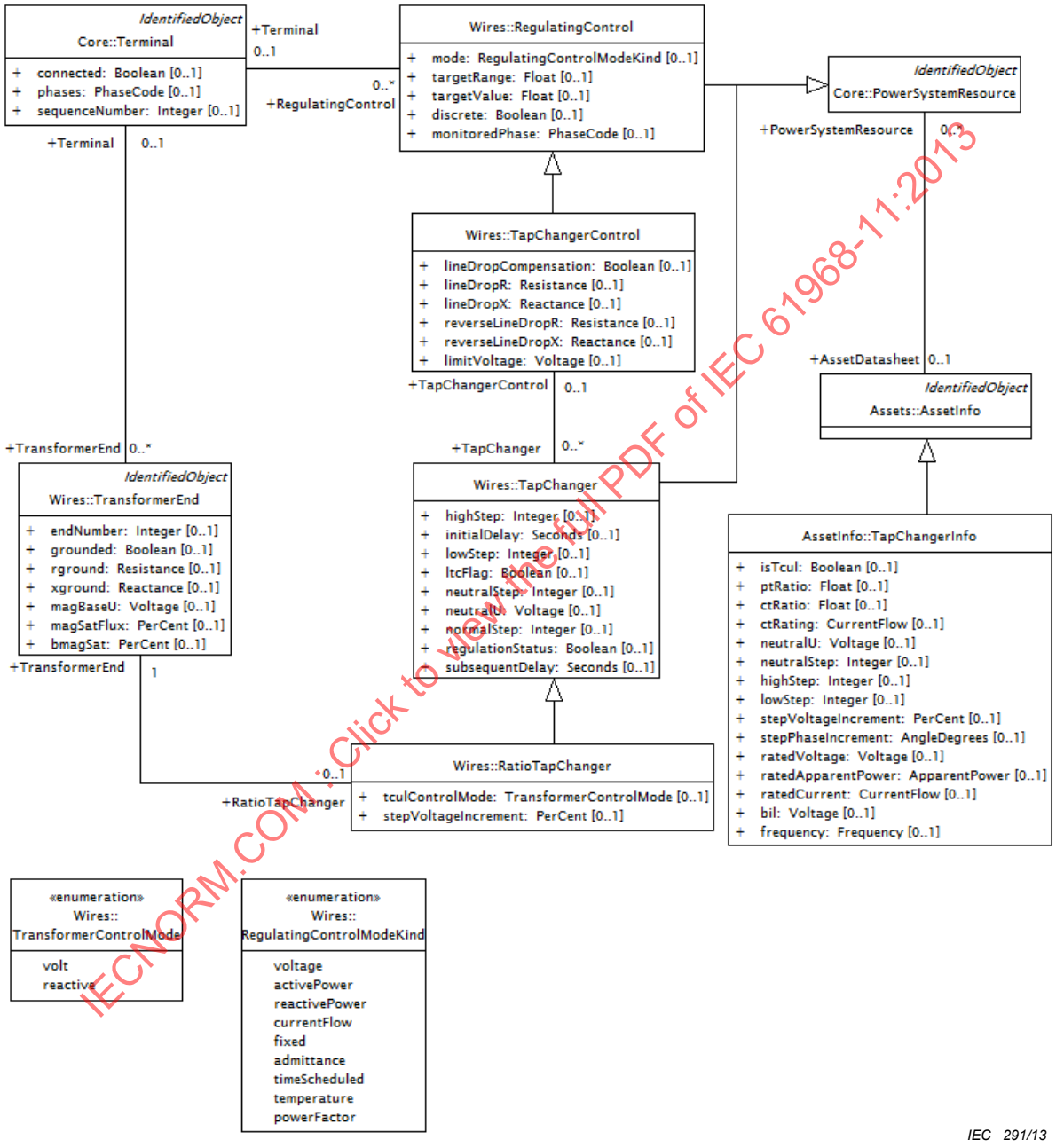


Figure 14 – DCIM tap changer model

A three-phase line voltage regulator usually has three independent regulators to help correct voltage unbalance. The regulators are usually connected in wye. The model starts with a **PowerTransformer** containing three **TransformerTank**s, and a total of six **TransformerTankEnd**s. There will be three instances of **RatioTapChanger**, each associated to a different **TransformerTankEnd**. The **RegulatingControl.monitoredPhase** attribute should be included among the **Terminal.phases** associated with the **PowerTransformer**. The tap positions, and sometimes the other attributes, will not be the same in each phase of the

regulator. An open-delta regulator is also fairly common; this consists of two single-phase regulators connected line-to-line in a bank, with partial capability to correct voltage unbalance.

A three-phase substation voltage regulator usually changes all three taps together, with no ability to correct voltage unbalance. In this case, tank-level modelling is not required and the model might consist of one **PowerTransformer** with two **PowerTransformerEnd**s. There would be just one **RatioTapChanger** associated to one **PowerTransformerEnd**. The **RegulatingControl.monitoredPhase** attribute can be **A**, **B**, or **C** if the potential transformer is connected line-to-ground. It can also be **AB**, **AC**, or **BC** for line-to-line potential transformers. Typically, only one potential transformer controls this type of regulator.

4.4.3.4.4 Example distribution transformer

The transformer in Figure 15 shows an open wye/open delta bank, which is used to supply inexpensive, three-phase service to smaller customers.

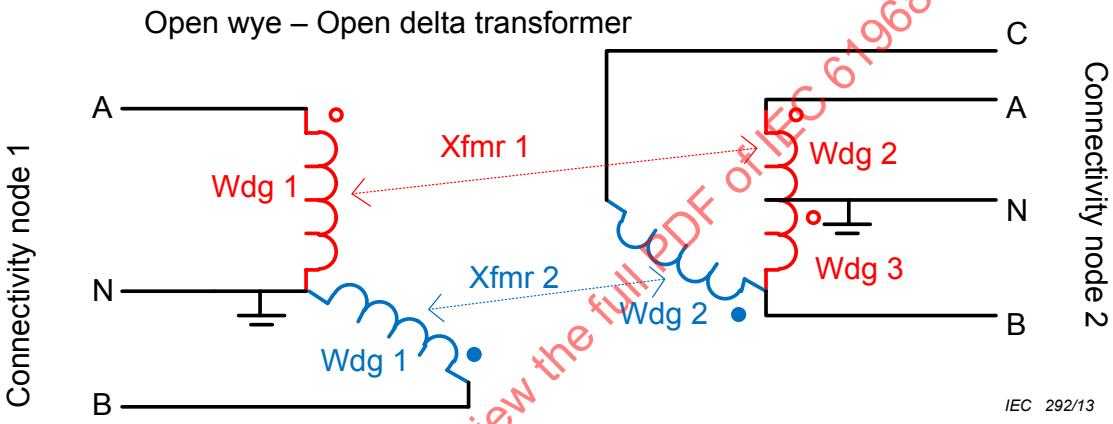


Figure 15 – Example of a distribution transformer that can be modelled with DCIM

Table 1 shows some of the important attribute values for this example. It requires tank-level modelling.

Table 1 – Open wye/open delta transformer bank connections

Transformer Tank	Transformer TankEnd	ratedU	ratedS	Connection Type	phaseAngle Clock	Transformer TankEnd. phases
Xfmr 1	Wdg 1	7 200	100e3	I	0	AN
	Wdg 2	120	50e3	I	0	AN
	Wdg 3	120	50e3	I	6	BN
Xfmr 2	Wdg 1	7 200	50e3	I	0	BN
	Wdg 2	240	50e3	I	0	BC

A phase angle clock value of “6” indicates that Wdg 3 is actually from N to B, rather than B to N. Through the **Terminal**s, **ConnectivityNode** 1 will have phases **ABN** present. Other connected equipment, such as a line segment, could add phase **C**. **ConnectivityNode** 2 will have phases **ABCN** present. The “lighting leg” (Xfmr 1) usually has a different rating than the “power leg” (Xfmr 2). This means that phase and rating assignments to the bank might be ambiguous, and thus need to be specified on **TransformerTankEnd**.

4.4.3.4.5 Example autotransformer

An autotransformer is made by connecting two transformer windings in series, so there is a metallic connection between the two voltage levels. The main advantage is a cost savings, because the MVA rating is higher than for the same two windings connected as a conventional transformer. Autotransformers are also more efficient and have less voltage drop. The main disadvantage is probably higher short circuit current in fully developed systems, because autotransformers have lower impedance. Two common applications are:

- Transformations between two extra-high voltage (EHV) levels in a substation, where the cost savings are important for turns ratios up to approximately 2:1.
- Line voltage regulators on distribution feeders, where the regulating (buck/boost) winding is connected in auto.

Figure 16 shows a two-winding transformer to the left, with a short-circuit test connection on the L winding terminal. All of the current is transformed magnetically, and the turns ratio is n_1:n_2. To the right, the same two windings are connected as an autotransformer. The red current flows directly to the short-circuit test connection on the L winding terminal. In addition, the magnetically transformed blue current also contributes to the short-circuit current, which is higher than in the two-winding case. The autotransformer turns ratio is (n_1+n_2):n_2. The H winding is sometimes called the series (S) winding in an autotransformer, and the L winding is sometimes called the common (C) winding in an autotransformer.

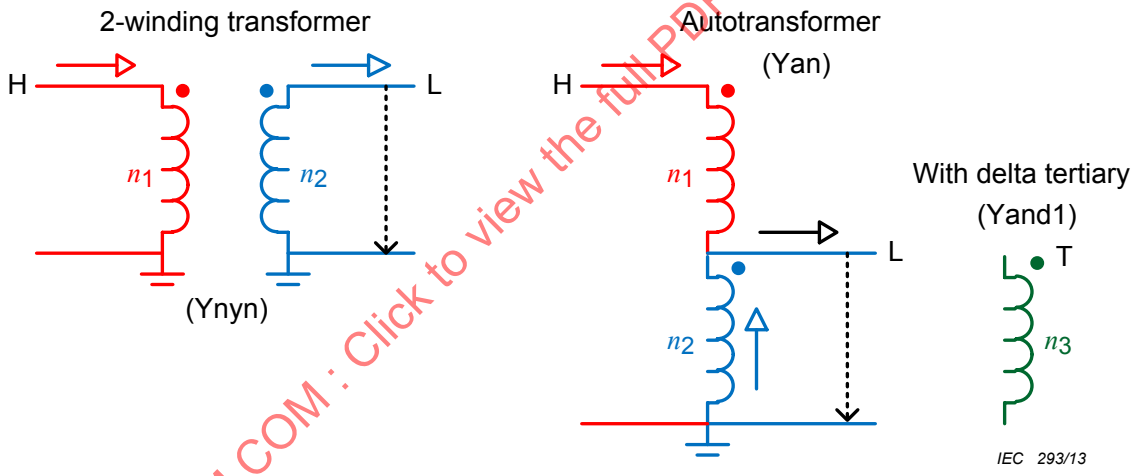


Figure 16 – Example of a two-winding transformer connected as an autotransformer

The IEC 60076-1 vector groups define grounding and phase shift characteristics of each winding, which are important for protective relaying, paralleling, and other applications – Figure 16 shows these in parentheses. “Ynyn” denotes a transformer with two windings, both wye grounded. (External neutral impedance may still be added, but is not listed in the vector group). The autotransformer could also be denoted “Ynyn”, because it has the same neutral connection and phase shift characteristics. However, some transformer vendors use “A” or “a” to denote an autotransformer winding. Figure 16 shows the “Yan” vector group for an autotransformer. Even though the H terminal has a conducting path to neutral through the L winding, the H winding itself is not connected to the neutral. As per IEC 60076-1, the highest voltage winding is capitalized in the vector group, while all other windings are lower-case. In practice, this means the “a” for an autotransformer would always be lower case.

Many autotransformers have a delta tertiary, shown to the right of Figure 16. The vector group would be “Yand1”, where the 1 refers to a 30° lag (1 o’clock) with respect to the H winding. The Z_{HT} mesh impedance value is affected by the auto connection, but not the Z_{LT} mesh impedance value. The effect on Z_{HT} is not presented here, but may be found in several technical references.

The conversions from two-winding data (left side of Figure 16) to autotransformer turns ratio (N), volt-ampere rating (S_{auto}) and mesh impedance (Z_{auto}) on the right side of Figure 16 are:

$$N = 1 + n_1 / n_2$$

$$S_{auto} = S_{2\text{-wdg}} [N / (N - 1)]$$

$$Z_{auto} = Z_{2\text{-wdg}} [(N - 1) / N]^2$$

In per-unit, the converted autotransformer impedance ($Z_{pu\text{-}auto}$) is:

$$Z_{pu\text{-}auto} = Z_{pu\text{-}2\text{-}wdg} / N$$

For example, suppose the 2-winding transformer is 115/115 kV, rated 100 MVA, with 10 % impedance on 100 MVA. The turns ratio is 1:1. Viewed from either winding, the short-circuit impedance is 13,225 Ω. Connected as an autotransformer, 230/115 kV, the turns ratio is 2:1 (N=2) and the rating is 200 MVA. Viewed from the 115-kV terminal, the short-circuit impedance is 3,306 25 Ω, which is 5 % on the new rating of 200 MVA. The iron core and copper winding costs are half of what they would be for a conventional 2-winding transformer of the same rating. The total cost is somewhat more than 50 %, because the L winding leads must carry more current, and the H winding must be insulated for a higher voltage. The test sheet for this autotransformer would show a voltage ratio of 230 / 115 kV, a rating of 200 MVA, and a short-circuit impedance of 5 %.

It is common to model an autotransformer as a conventional two-winding transformer, using data from the test sheet, ignoring the fact that the windings are actually connected in series. However, there are times when the difference is important, such as more accurate core modelling, or more accurate modelling of the impedance vs. tap characteristic. It is possible to derive the physical autotransformer model in Figure 16, if the series and common windings have been identified.

In the CIM, an autotransformer should be modelled with conventional two-winding data for impedances, admittances, and ratings, as typically found on autotransformer test reports. Each physical winding will have a corresponding **PowerTransformerEnd** or **TransformerTankEnd** in the CIM. It is optional to specify the autotransformer connection with an attribute value **A** for **connectionKind** on the common end, and with “an” appearing as part of **PowerTransformer.vectorGroup** for that end. The series end shall then have attribute value **Y** for **connectionKind**. The series and common **endNumber**s should be 1 and 2, respectively. The receiving application may then derive the physical autotransformer model if needed. To ignore autotransformer connections in the model, specify **Yn** for **connectionKind** on both series and common ends; there is no restriction on the **endNumber**s. The **connectionKind** attribute appears on **TransformerEndInfo** if using tank-level modelling and on **PowerTransformerEnd** if not using tank-level modelling. Note that **A**, **Y**, **D**, and **Z** are always capitalized in **connectionKind**, but whenever the **endNumber** is greater than 1, they should be lower-cased in the **vectorGroup**.

4.4.3.5 Auxiliary equipment

This edition of IEC 61968-11 includes the means to model auxiliary equipment through the **IEC61970::Wires** model package from IEC 61970-301, shown in Figure 17.

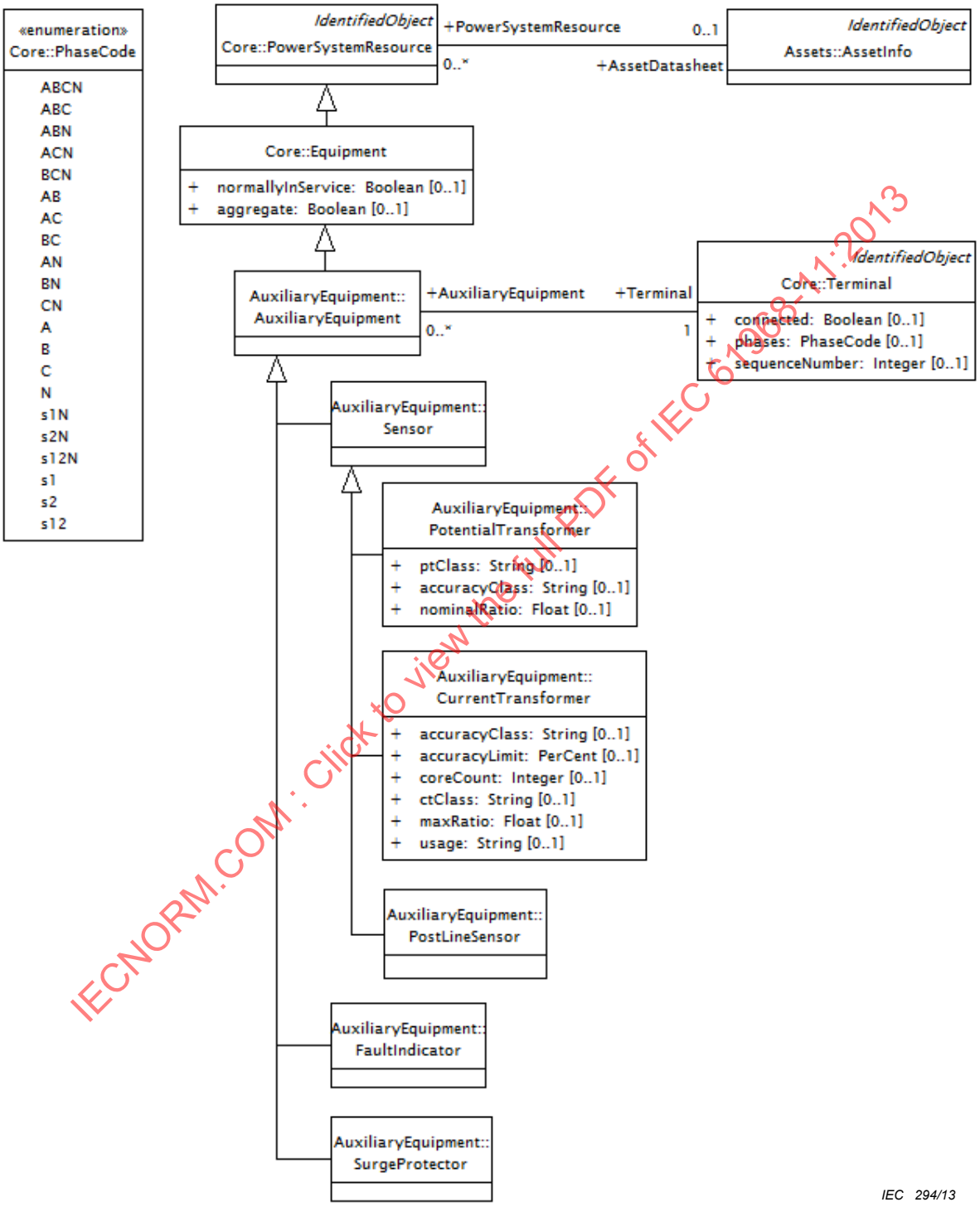


Figure 17 – DCIM auxiliary equipment

An important addition is the association of any **AuxiliaryEquipment** with a **Terminal**, which allows for positioning of the auxiliary devices in the electrical network connectivity model. For instance, it is now possible to explicitly model instrument transformers, position them on single line diagrams along with the usual **ConductingEquipment** and deduce the **Measurement**s

that they actually produce. This kind of data exchange would be typical when importing substation models defined with other data models than CIM.

Also, it is now possible to exchange the "placement" in the network of **FaultIndicator** instances, as used for fault location, isolation and restoration purposes to assist with the dispatch of crews to the part of the network (**ACLineSegment**) where the fault most likely happened.

The addition of the **Sensor** class provides the point of extensibility for other kinds of sensing equipment than the instrument transformers and **PostLineSensor**.

NOTE In this part of of IEC 61968 (documenting DCIM11), there is no support for datasheet modelling of any of these auxiliary devices, i.e., Figure 18 shows the **AssetInfo** class without specialisations. It is planned to add the corresponding assets model classes in the next edition, to allow for more detailed description of these assets in support of data exchanges required by IEC 61968-3 and IEC 61968-4.

4.4.4 Customers model

The **Customers** package introduces classes, shown in Figure 18, that are needed for the enterprise integration of customer information and billing systems and the information they exchange with other enterprise systems.

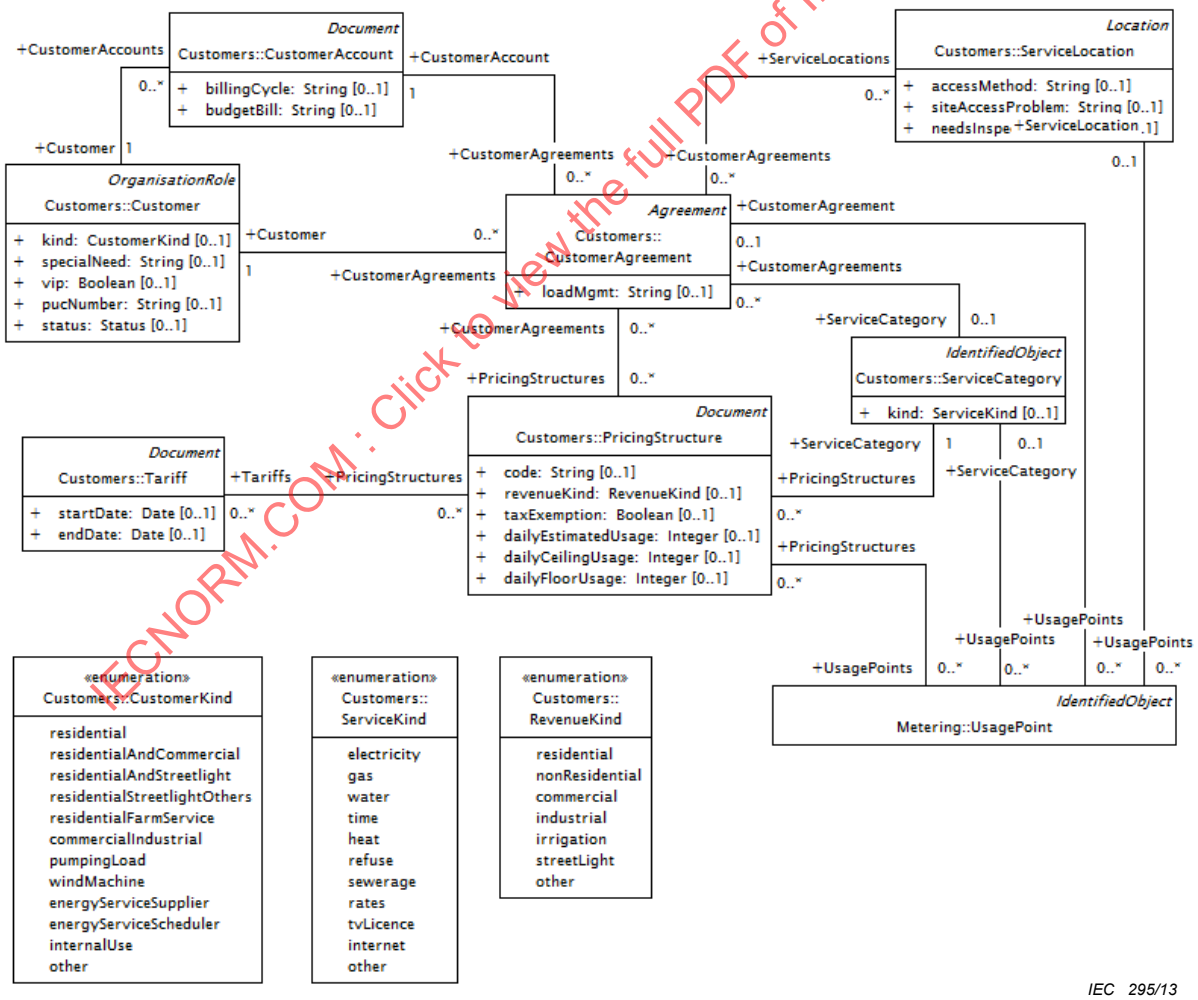


Figure 18 – DCIM customers model

A **Customer** is an organisation receiving services from one or more **ServiceSupplier**s. A **Customer** may have multiple **CustomerAccount**s and/or **CustomerAgreement**s.

A **CustomerAccount** is the mechanism for customer billing and payment. It is used to create billings (invoices) for a **Customer** and to receive payment. A **CustomerAccount** will typically be billed on a defined schedule/billing cycle. A **CustomerAccount** can support the aggregate billing across multiple **CustomerAgreement**s. Thus there may be multiple **CustomerAgreement**s linked to a given **CustomerAccount**.

A **CustomerAgreement** is the agreement between the **Customer** and the **ServiceSupplier** to pay for service at a specific service delivery point (**UsagePoint**, see metering model in 4.4.5.1). It identifies certain billing information used during **Charge** creation, including the **PricingStructure**.

A **PricingStructure** is a grouping of pricing components and prices used in the creation of customer **Charge**s and the eligibility criteria under which these terms may be offered to a **Customer**. The reasons for grouping include regulatory jurisdiction, customer classification, site characteristics, classification (i.e. fee price structure, deposit price structure, electric service price structure, etc.) and accounting requirements. A **PricingStructure** can embody one or more **Tariff**s which are often referred to as rate schedules.

CustomerAccount, **PricingStructure** and **CustomerAgreement** are all specialisations of **Document**, either directly or through the **Agreement** class. This demonstrates the intended usage of the **Document** class by specialising it as appropriate, and never using the **Document** class directly for relationships.

A **ServiceCategory** is the category (electric, water, gas, etc.) of service provided. A **ServiceSupplier** is the organisation that provides services to customers. There are multiple types of **ServiceSupplier**s (for e.g. utility, retailer, or other). A given **ServiceSupplier** can provide services in multiple instances of **ServiceCategory**.

A **ServiceLocation** is a real estate location, commonly referred to as the premises to which one or more services may be delivered. It may have multiple **UsagePoint**s; however, each **UsagePoint** will be for one and only one **ServiceCategory**. Additionally, a **ServiceLocation** may have multiple **UsagePoint**s for a given **ServiceCategory**.

NOTE This part of IEC 61968 (documenting DCIM11) provides customer modelling for the needs of IEC 61968-9 only. The next editions will present more comprehensive model in support of IEC 61968-8 and IEC 61968-6, as well.

4.4.5 Metering model

4.4.5.1 General

The **Metering** package introduces classes essential to the enterprise integration of metering systems and the information and control commands they exchange with other enterprise systems.

Figure 19 illustrates classes that are specific to the metering domain;[15] refer also to Figure 18 in 4.4.4 for customer-related classes, also relevant in the metering domain.

[15] Because of the number of classes contained, only their names and relationships are shown in the figure; 4.4.5.2 to 4.4.5.7 zoom into subsets of related classes.

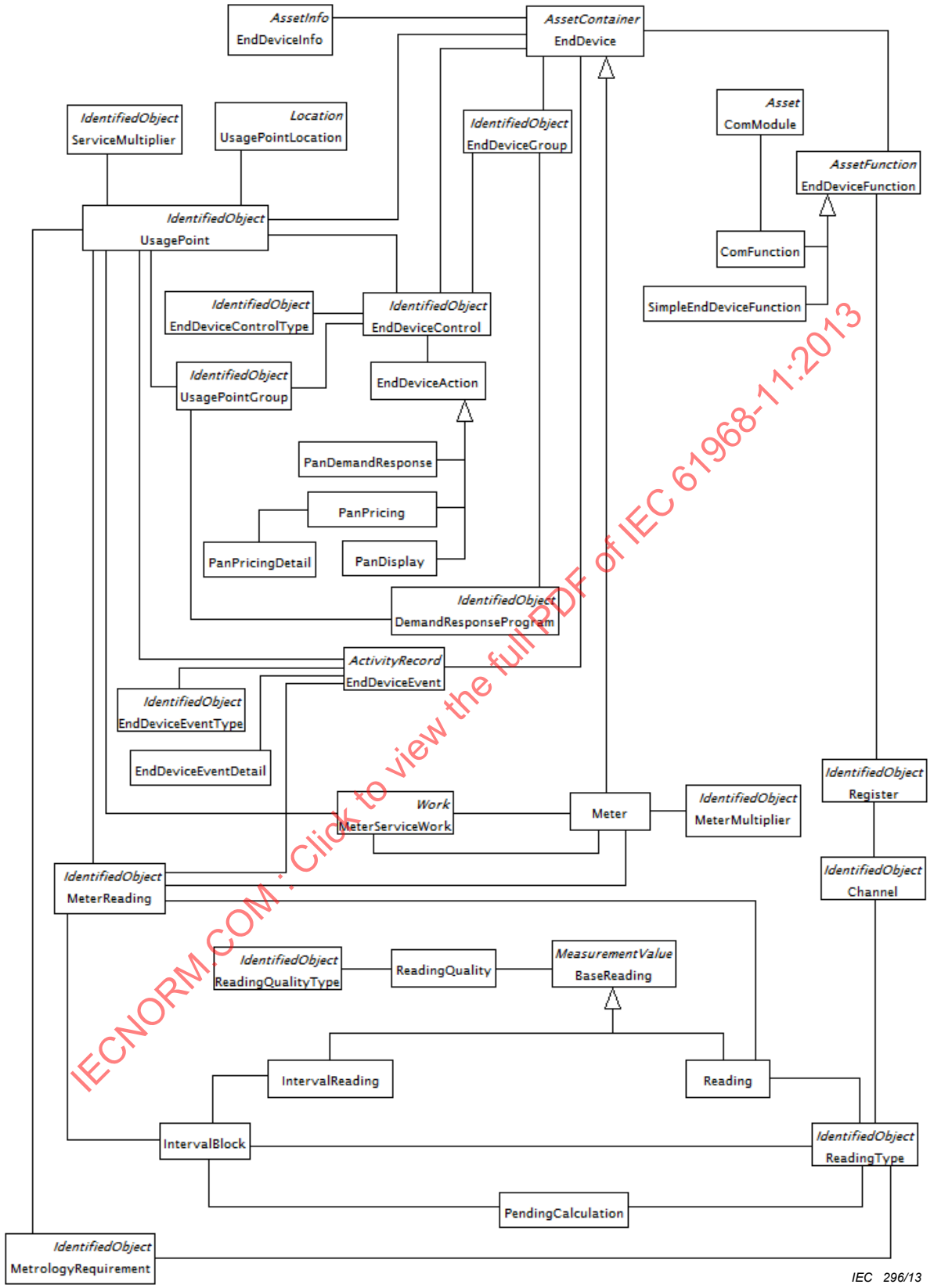


Figure 19 – DCIM metering model

These classes support typical metering domain entities, as well as concepts required for operational data exchanges. 4.4.5.2 to 4.4.5.7 describe related subsets in more detail. 4.4.5.2 through 4.4.5.4 focus mainly on the domain entities that typically need to be configured in

order to enable operational data exchanges. These types of operational data exchanges are described in greater detail within 4.4.5.5 through 4.4.5.7.

4.4.5.2 Usage points

One of the cornerstone concepts in the metering model is the **UsagePoint**, shown with its relationships in Figure 20.

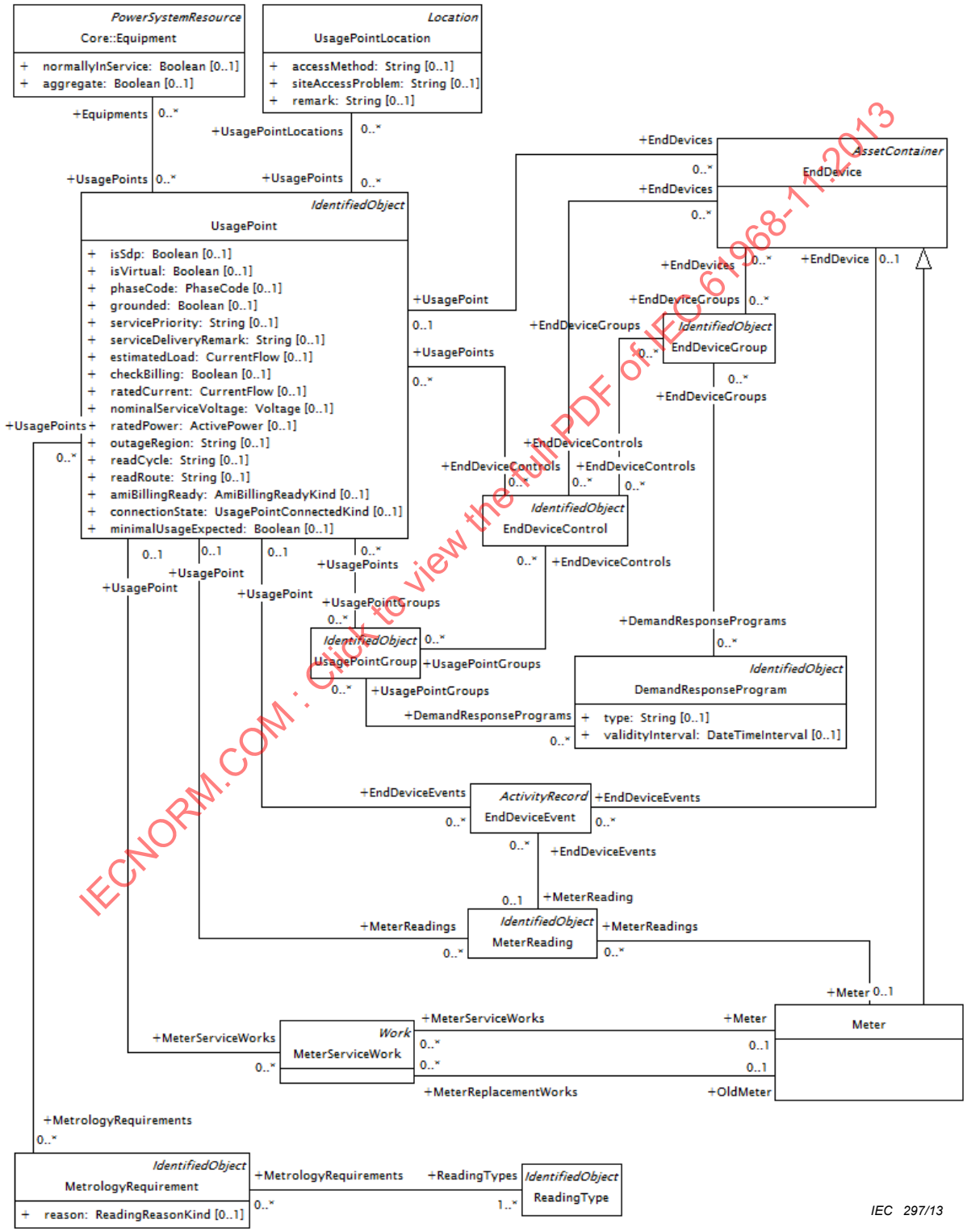


Figure 20 – DCIM usage point model

The **UsagePoint** is the logical or physical point in the network to which **MeterReading**s or **EndDeviceEvent**s may be attributed. It is used at the place where a physical or virtual **Meter** or other **EndDevice** may be located; however, it is not required that the device be present.

A **UsagePoint** can be a service delivery point which is defined as the point of demarcation in a network where ownership transitions from **ServiceSupplier** to **Customer** (see Figure 18).

A **UsagePoint** may have zero or more **MetrologyRequirement**s which define the **ReadingType**s that must be acquired or calculated for the **UsagePoint**.

UsagePointLocation defines the physical location of **UsagePoint**s.

MeterServiceWorks are dynamically associated with a **UsagePoint**, and provide for an explicit specification when the work consists in meter replacement, through its **OldMeter** association end.

UsagePoint is the class providing the link between the network-oriented model of electrically connected equipment on the one hand, and the asset and premises-oriented model of the metering domain on the other. That linkage can be realised through the relationship of **UsagePoint** with **Equipment**, the class defined in the **IEC61970::Wires** model package from IEC 61970-301. This relationship allows for exchanging the configuration of relationships between at least the following concepts in the two domains:

- A **TransformerTank** supplies power to one or more connected **UsagePoints**. The knowledge of this relationship is required by certain outage management and distribution network management functions.
- One or more **UsagePoint**s may represent aggregated **EnergyConsumer**s, as seen by typical control room systems.
- Instrument transformers (**CurrentTransformer** and **PotentialTransformer**) physically installed at a **UsagePoint** may be explicitly modelled through this relationship, and their datasheet information can be defined in one system and shared with the other systems.

UsagePoints may be dynamically grouped into multiple **UsagePointGroup**s, targeted at operations involving **MeterReading**s, **EndDeviceControl**s, and potentially other metering operations to defined groups of **EndDevice**s and/or **UsagePoints**. Messages between enterprise systems are utilised to synchronise each system's understanding of the membership of each group.

4.4.5.3 End devices

Another essential concept in the metering domain modelled in DCIM is that of **EndDevice**s, shown with its relationships in Figure 21.

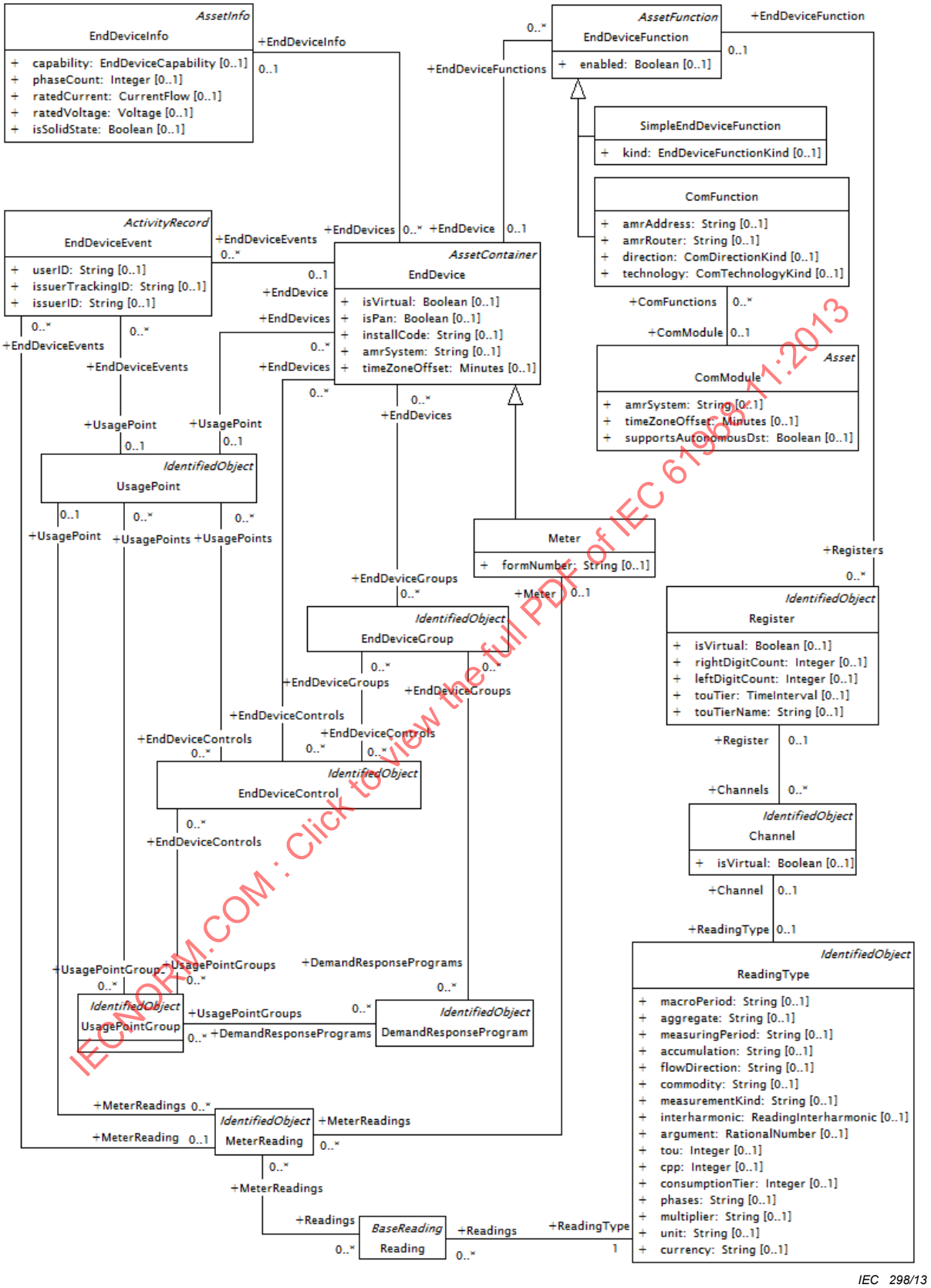


Figure 21 – DCIM end device model

An **EndDevice** is an **AssetContainer** that performs one or more **EndDeviceFunction**s. One type of **EndDevice** is a **Meter** which can perform metering, load management, connect/disconnect, and other functions. An **EndDevice** can also be used to model a premises area network (PAN) device, which may perform functions such as in-home display and control and monitoring of

premises-based equipment such as air conditioners, refrigerators, pool pumps, etc. One or more PAN devices can be linked to a **Meter** or a **UsagePoint**.

Multiple **EndDevice**s, thus also multiple **Meter**s, may be linked to a given **UsagePoint**. However, it is relatively common to encounter enterprise systems requiring a separate **UsagePoint** for each **Meter**.

EndDevices may have communication capability that is integral to the **EndDevice**. Also, the communication capability of **EndDevice**s may be provided by one or more **ComModule**s; this containment is inherited through the **AssetContainer**–**Asset** association.

A **Meter** can be a physical or virtual **EndDevice** that performs the metering function. A physical meter exists when there is real world hardware that performs the metering function. A virtual meter is realised in the absence of real world hardware and for example can be defined to perform functions such as aggregating the consumption for two or more physical meters.

Meters can have one or more **Register**s, which are devices that indicate or record units of a commodity or other quantity measured. Each **Register** in turn can have one or more **Channel**s. Each **Channel** is however reserved for one and only one **ReadingType**.

A **Channel** is a single path for the collection or reporting of a **Register**'s values over a period of time. For example, a **Register** which measures forward energy can have two **Channel**s: one providing bulk quantity readings and the other providing interval readings of a fixed interval size.

A **ReadingType** instance is used as a unique identifier that specifies the attributes required to fully characterize a **Reading**. A **Reading** is a specific value measured or calculated by a **Meter** or system.

NOTE Attributes of **ReadingType**, currently defined as strings in the UML model, will hold integer values defined in Annex C of IEC 61968-9:2009 in the data exchanges. It is the intention to model those values as enumeration literals in some of the next editions of IEC 61968-11. Harmonisation will be required with the equivalent enumeration types defined in the **IEC61970::Domain** model package from IEC 61970-301, for the following attributes of **ReadingType**: **phases**, **multiplier**, **unit** and **currency**.

4.4.5.4 Configuration events

In contrast to electrical network models, where there is usually one system that is the master data source, in the metering domain there may be several systems of record, each being a master for only a subset of business entities. Therefore, tracking of configuration changes requires high flexibility and is therefore modelled with **ConfigurationEvent** instances, as shown in Figure 22.

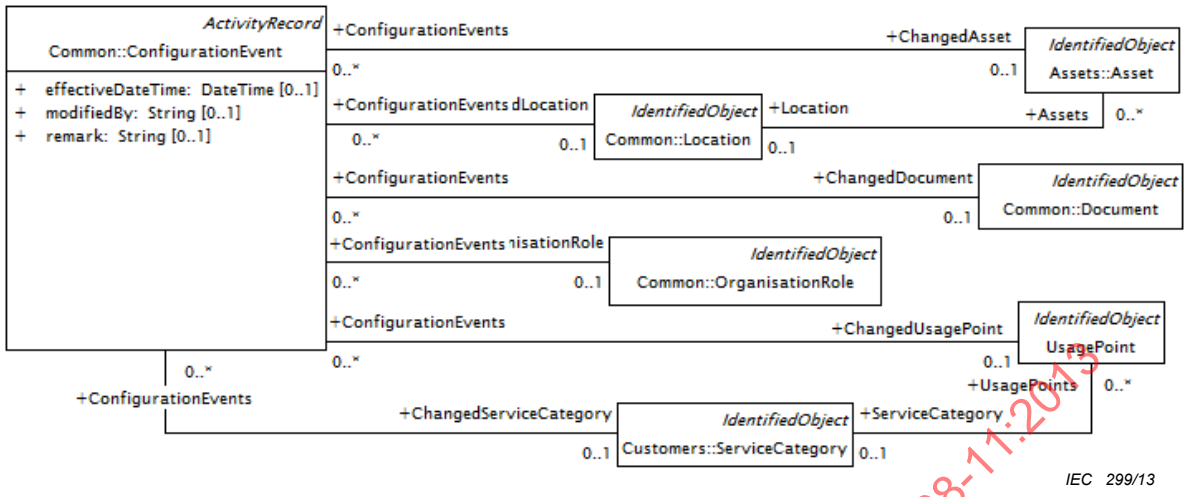


Figure 22 – Configuration events for metering

ConfigurationEvent is a specialisation of **ActivityRecord**, used exclusively to exchange data about configuration changes for various objects: **Asset**s, **Location**s, **Document**s, **OrganisationRole**s, **UsagePoint**s and instance of **ServiceCategory**.

4.4.5.5 Meter readings

Meter readings are exchanged between enterprise systems to support functions such as billing, demand response, load management, system planning, and revenue protection. Figure 23 illustrates classes that support this kind of data exchange.

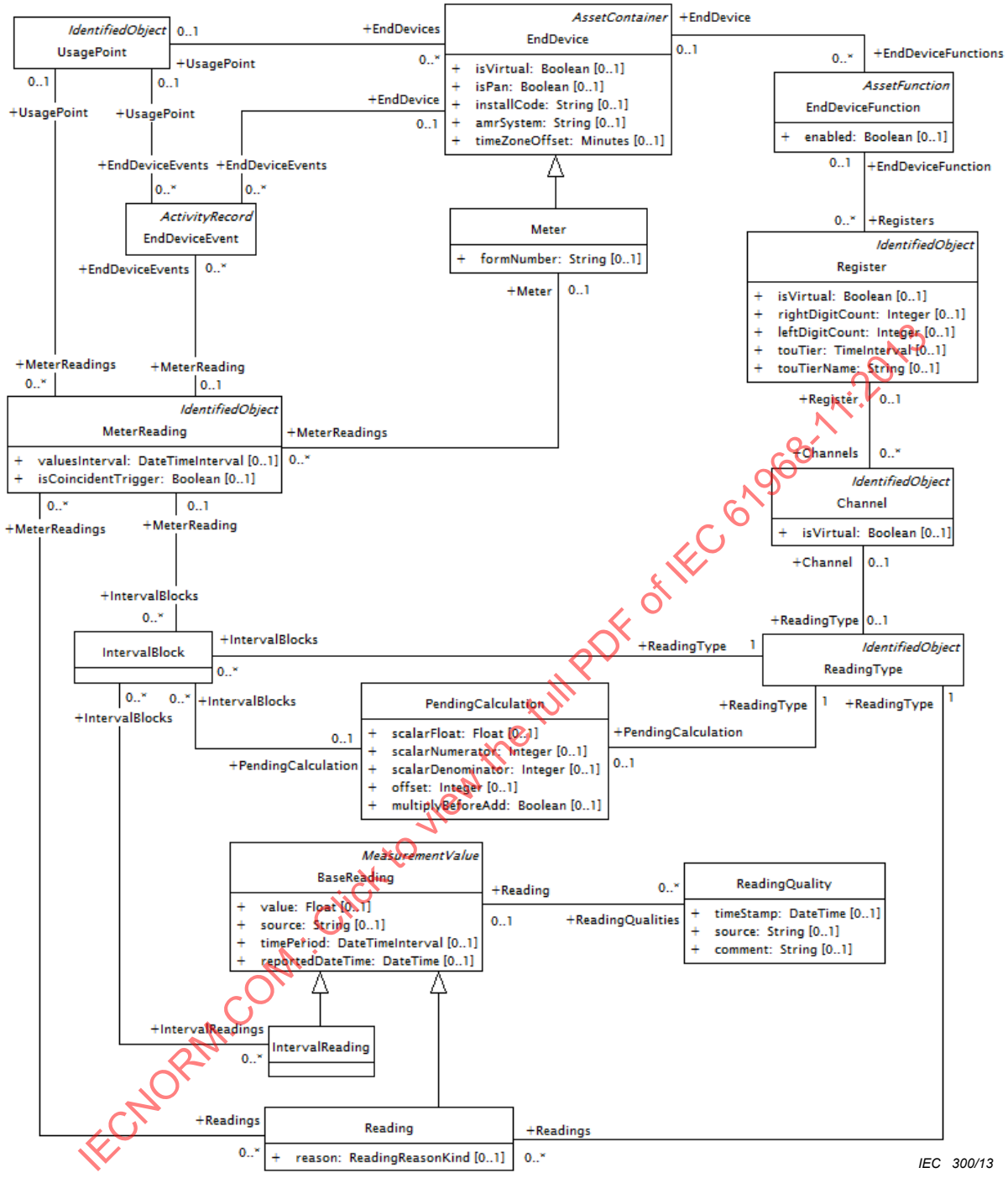


Figure 23 – DCIM meter readings model

MeterReadings may consist of various combinations of **Reading**s and **IntervalBlock**s. **Reading**s represent values (either measured or calculated) at a specific point in time and that are characterised by a **ReadingType** and one or more instances of **ReadingQuality**. **IntervalBlock**s represent collections of **Reading**s at equally spaced time intervals.

A typical use case for **Reading**s is that of a metering system or meter data management system providing a billing system with midnight register readings. **IntervalBlock**s can be used to provide a customer information system with finer grained consumption readings, for example for 15-min or hourly periods. The latter may be needed directly for billing or to apportion consumption into time-of-use periods which may be billed at different rates.

NOTE Although both **Reading** and **IntervalReading** specialise the class **MeasurementValue**, defined in the **IEC61970::Meas** model package from IEC 61970-301, there is very little in common between the two models for measurements. The only attribute from **MeasurementValue** used is **timeStamp**; none of its association ends is used. Even if it is possible to associate a **Reading** with an instance of **Measurement**, source, value, quality and type definitions are completely different. Therefore, for applications or systems that would like to process a **Reading** as a **MeasurementValue**, or vice versa, the standard DCIM currently does not provide any support for transformation between the two representations.

4.4.5.6 End device controls and events

Certain end devices are capable of performing actions in response to control, or to detect the occurrence of particular events and to capture and transmit to other systems specific details about these events. DCIM model classes supporting this kind of data exchange are illustrated in Figure 24.

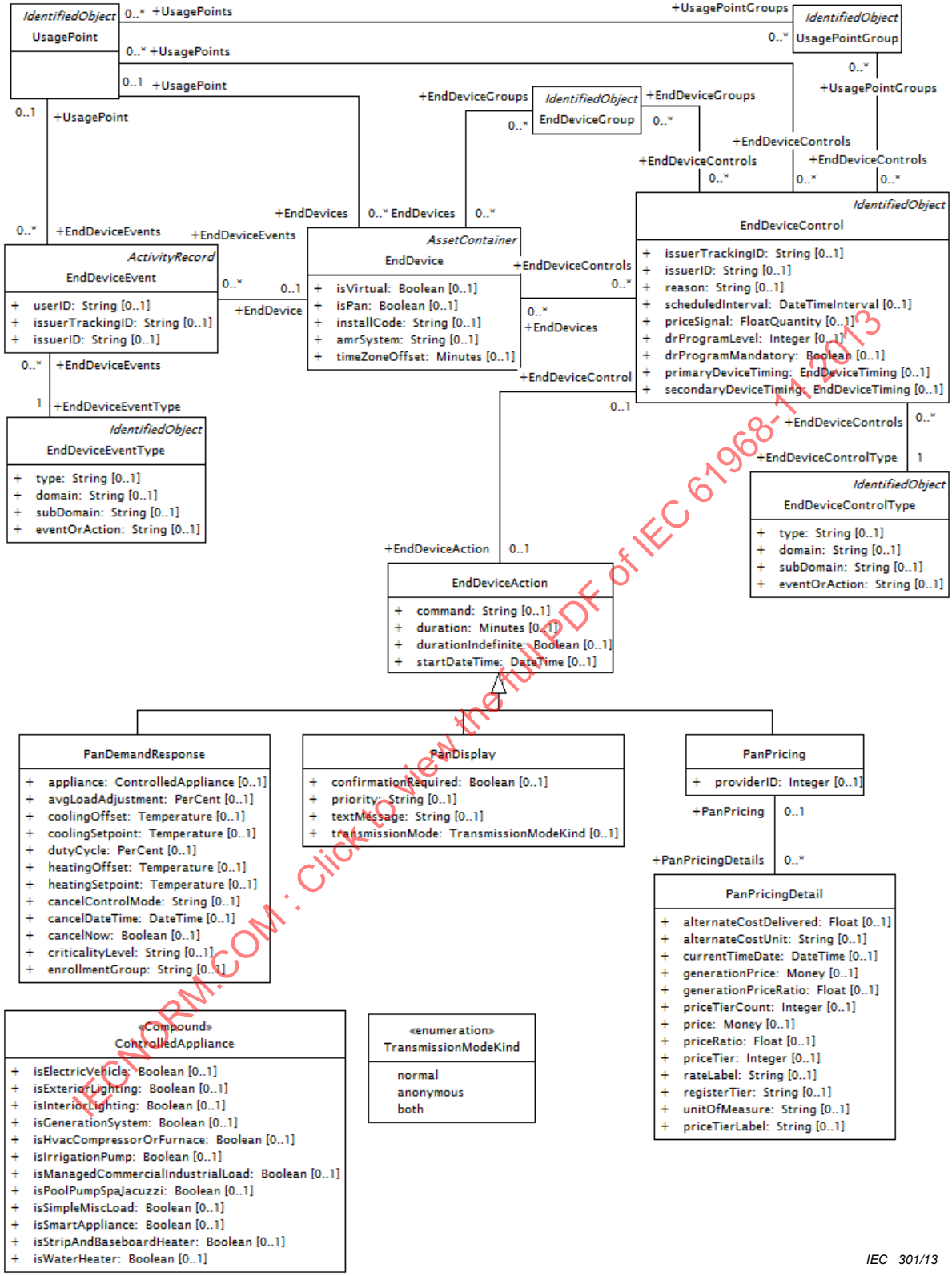


Figure 24 – DCIM end device controls and events model

An **EndDevice** may be capable of acting upon **EndDeviceControl** commands. Examples include automated connect or disconnect of service by a **Meter**, resetting the demand register of a **Meter**, displaying a text message on a PAN device, communicating pricing information to a PAN device or other **EndDevice**, etc.

EndDeviceControl commands are typically requested by enterprise systems such as a customer information system, and ultimately are passed to the metering system for execution. An **EndDeviceControl** specifies the desired control action using an **EndDeviceControlType**. The enterprise system may request a control for immediate execution or execution at a later date and time.

Systems may target **EndDeviceControl**s to individual **EndDevice**s (or **Meter**s) or **UsagePoint**s. Alternately, they may target one or more **EndDeviceGroup**s or **UsagePointGroup**s.

Certain **EndDevice**s may be programmed to control other equipment which is not the direct subject of the **EndDeviceControl**s. For example, a PAN device may be sent the details of a demand response event (using an **EndDeviceControl**) which will ultimately result in the switching on and off of equipment such as air conditioners, pool pumps, etc. For this data exchange, the information required to "program" the PAN device is communicated in the **EndDeviceControl** with an associated **EndDeviceAction** (which is further specialised as **PanDemandResponse**, **PanPricing**, or **PanDisplay**).

Figure 24 shows also the classes relevant for modelling **EndDeviceEvent**s; they are detected and captured by **EndDevice**s and transmitted to other enterprise systems. **EndDeviceEvent**s frequently originate in a metering system (which is in direct communication with the **EndDevice**) and are transmitted to other systems as trigger events for the execution of various business logic actions. Examples of **EndDeviceEvent**s include loss or restoration of power, loss or restoration of communication functions, meter tampering alarms, reverse rotation flags, and many, many others. For example, power loss alarms could be communicated to an outage management system for diagnosis of outage extent. Or, reverse rotation and tampering alarms could be transmitted to a system responsible for detecting power theft or other revenue protection issues.

An **EndDeviceEvent** specifies the event using an **EndDeviceEventType**.

Events detected by **EndDevice**s may occur spontaneously (in an unsolicited manner) or as a direct result of an **EndDeviceControl** command. In the latter case they are often referred to as "consequential events". **EndDeviceEvent**s are typically associated to a **UsagePoint**.

4.4.5.7 Meter service requests

DCIM model also provides a basic support for data exchanges in the context of work management related to end devices. Relevant classes have been shown in Figure 20.

MeterServiceWork is a specialisation of **Work**, where an **EndDevice** is involved.

For example, it may be necessary to install, remove or configure **Meter**s as a consequence of the registration of a new **Customer**, removal of a **Customer** or the switch of a **Customer** from one **ServiceSupplier** to another. There may also be the need to change out a **Meter** which involves the removal of the old **Meter**, installation of the new **Meter** and configuration of the new **Meter** as needed by the metering system.

MeterServiceWork also provides support for collections of **MeterReading**s taken in conjunction with the service work being performed.

4.4.6 Payment metering

4.4.6.1 General

Payment metering is an extension of the **Metering** model package and contains the information classes that support specialised applications such as prepayment metering. These classes are generally associated with the collection and control of revenue from the customer for a delivered service.

4.4.6.2 Transacting

A payment metering system generally facilitates financial transactions between a customer and a service provider. The relevant information describing these transactions is typically recorded in the payment system and this information is subsequently exchanged with another system such as the customer information or billing system. A typical example of realising such an information recording scheme using some of the classes found in the **PaymentMetering** model package is shown in Figure 25.

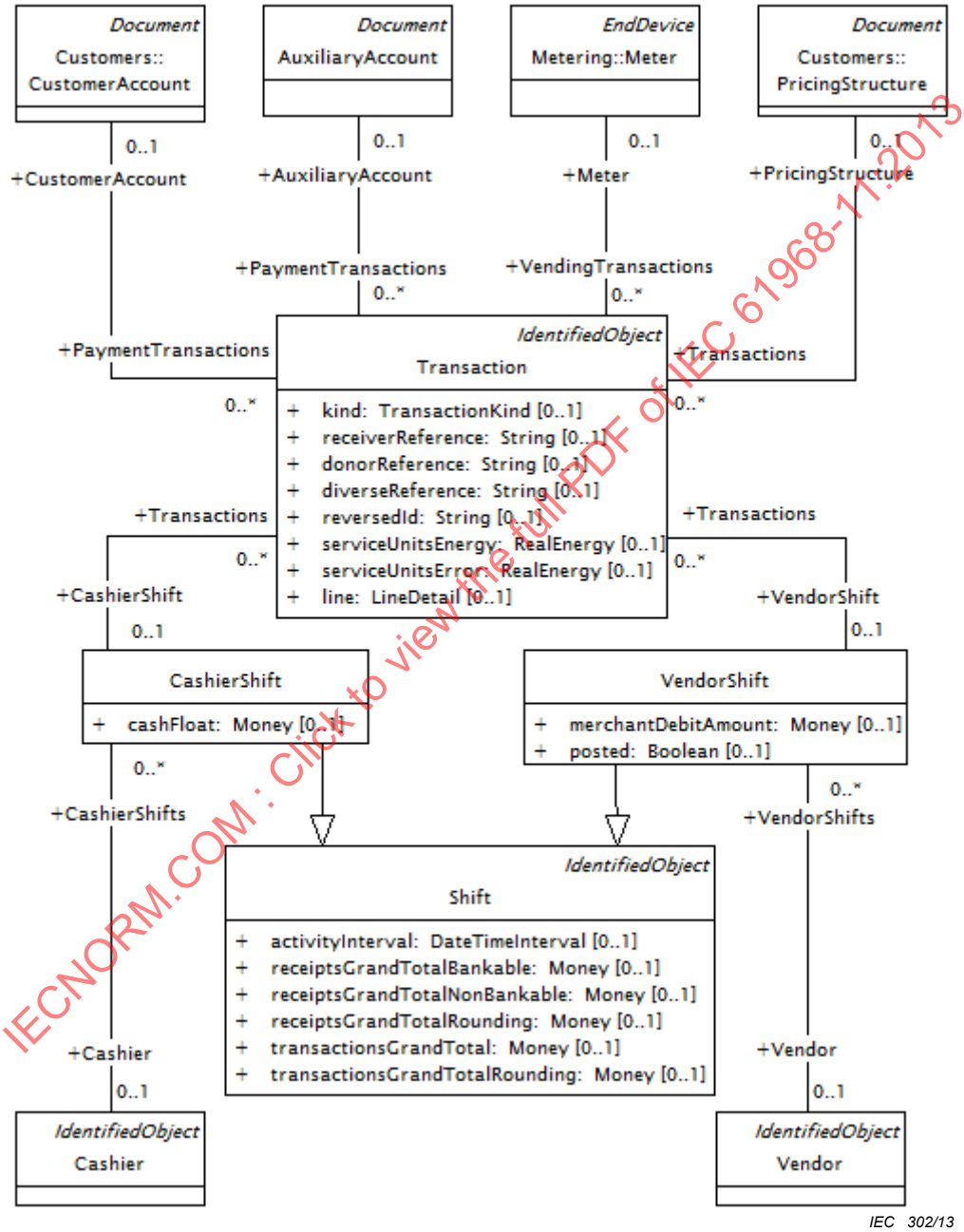


Figure 25 – DCIM transacting model

The core of information in this model is the **Transaction** class, which captures all the relevant information about the transaction and also includes extended information such as when a payment is made against a **CustomerAccount**, payment against an **AuxiliaryAccount**, purchase of a prepaid **Token** for a prepayment service meter and the pricing that was used to calculate the amount charged for such a sale.

Transaction information may be further aggregated and sorted into **CashierShift** and **VendorShift** groupings for accounting and reconciliation purposes against a **Cashier** and **Vendor** who are accountable for the revenue collected during the particular **Transaction**.

4.4.6.3 Receipting

A transaction generally involves the receipt of revenue from the customer, which may take the form of cash, cheque or card for example. The capture and subsequent exchange of information describing the properties of this revenue may be realised by means of the model example shown in Figure 26.

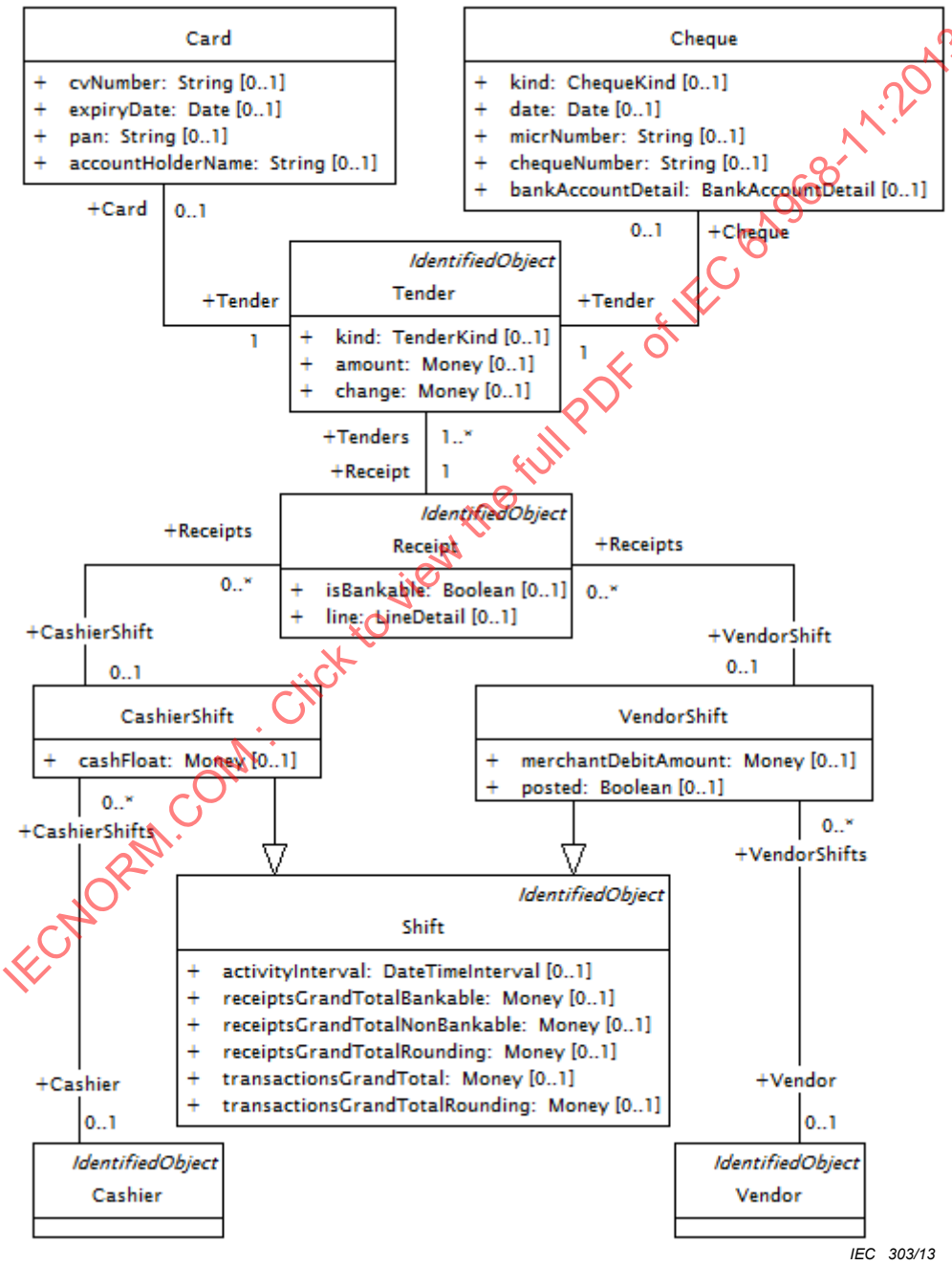


Figure 26 – DCIM receipting model

When a customer tenders payment during a transaction, the information is typically captured in the **Receipt**, **Tender**, **Card** and **Cheque** classes.

Receipt information may be further aggregated and sorted into **CashierShift** and **VendorShift** groupings for accounting and reconciliation purposes against a **Cashier** and **Vendor** who are accountable for the revenue collected during the particular **Transaction**.

4.4.6.4 Auxiliary payments

In addition to the typical payments made by customers for services provided by the service provider such as a utility, it is often required to receipt payments for other items such as debt, rates, taxes, municipal fines, TV licences, garbage collection charges, etc. The collection of such revenue may be integrated with token sales and customer account payments by means of auxiliary agreements and auxiliary accounts, an example of which is shown in Figure 27.

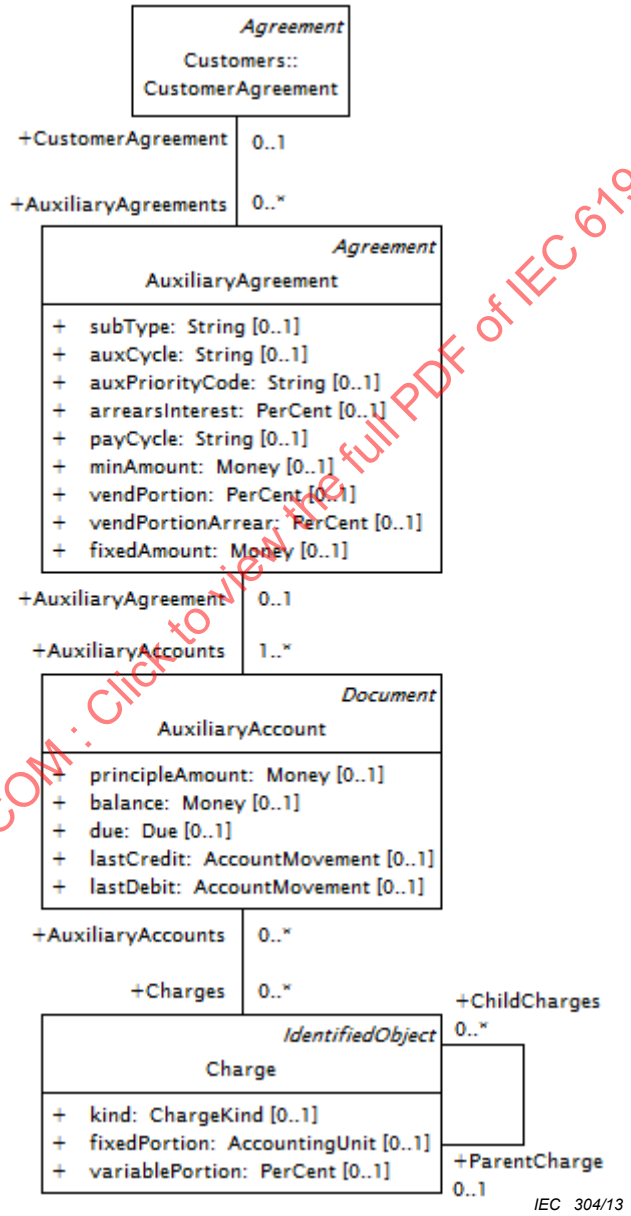


Figure 27 – DCIM auxiliary agreement model

AuxiliaryAgreement essentially extends from **CustomerAgreement** and captures the static rules of how the auxiliary account is managed. It captures the dynamic information about the charges and payments made against the account.

Charge allows for nested structures of charges to be levied against the **CustomerAccount** in accordance with the rules set in **AuxiliaryAgreement** and allows for fixed charges, variable charges and percentage charges.

4.4.6.5 Pricing and tariff structures

Pricing structures may contain tariffs, which are often quite complex in structure and operation. Most tariffs levy charges that are time-based or consumption-based, both of which are interval-based. A model to realise such complex tariff structures is shown in Figure 28.

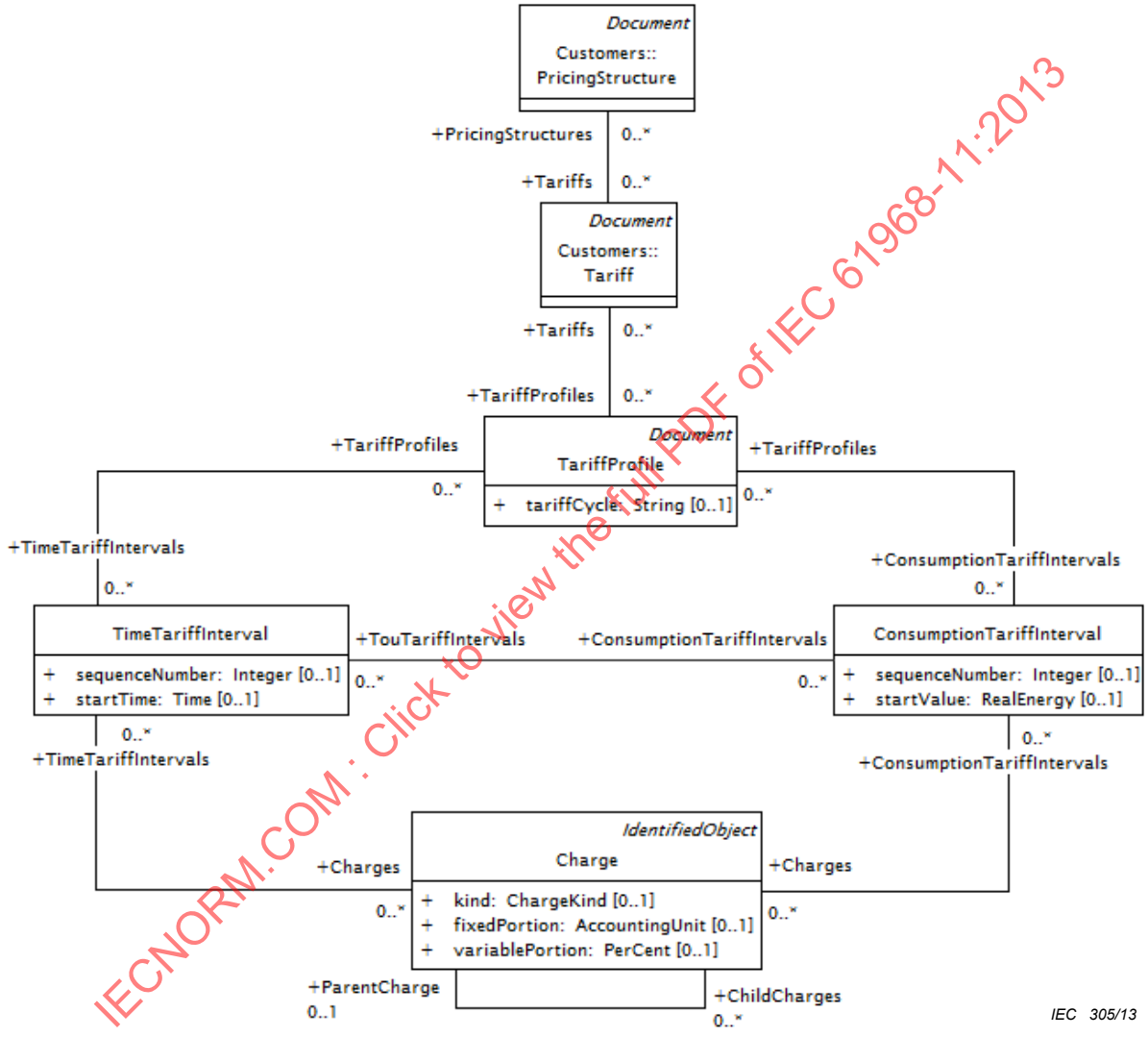


Figure 28 – DCIM pricing structure model

TariffProfile determines the cycle of operation for the **Tariff**, such as hourly, daily, weekly, monthly, etc, at the end of which it resets to start at the beginning of the process again.

TimeTariffInterval determines the starting time for a particular interval and several instances of **TimeTariffInterval** may be used to construct a series of time intervals to realise a time of use tariff for example.

Alternatively **ConsumptionTimeInterval** determines the starting value of a consumption interval and several instances of **ConsumptionTimeInterval** may be used to construct a series of consumption intervals to realise a block tariff or a step tariff for example.

The price per service unit per time interval or per consumption interval is determined by the **Charge** class, which provides for nested charge structures and allows for fixed charges, variable charges and percentage charges.

For very complex tariff structures, the **TimeTariffInterval** and **ConsumptionTimeInterval** may be combined to provide for time-based and consumption-based charges simultaneously.

4.5 Other

4.5 to 4.8 of IEC 61970-301:— describe CIM modelling tools, CIM extensions, and implementation conventions.

5 Detailed model

5.1 Overview

The common information model (CIM) represents a comprehensive logical view of information exchanged among different systems in electrical utilities. This definition includes the public classes and attributes, as well as the relationships between them. 5.2 describes how Clause 6 is structured. Clause 6 is automatically generated from the CIM model maintained with the tools described in 4.6 in IEC 61970-301:—.

5.2 Context

The CIM is partitioned into subpackages. Classes within the packages are listed in the order they are defined in the UML model. Native class attributes are listed first, followed by inherited attributes in order of depth of inheritance. Native associations are listed first for each class, followed by inherited associations in order of depth of inheritance. The associations are described according to the role of each class participating in the association. The association ends are listed under the class at each end of an association.

Figure 1 shows that distribution CIM (this document) depends on base CIM (IEC 61970-301). This document includes the detailed description of the contents of **IEC61968** package only, and references several classes, attributes and association ends included in **IEC61970** package.

For each package, the model information for each class is fully described. Attribute and association end information for native and inherited attributes is documented as in Table 2 and Table 3. For any inherited attributes or association ends the “description” column will contain text indicating the attributes are inherited from a specific class. The description column for native attributes and association ends contains the actual description.

Table 2 – Attribute documentation

name	type	description
native1	Float	A floating point native attribute of the class is described here.
native2	ActivePower	Documentation for another native attribute of type ActivePower.
name	String	inherited from: IdentifiedObject

In the attribute documentation (Table 2), in some cases, an attribute is a constant, in which case the phrase “(const)” is added in the name column of the attributes table. In such cases the attribute normally has an initial value also which is preceded by an equal sign and appended to the attribute name.

Table 3 – Association ends documentation

[mult from]	[mult to] name	type	description
[0..1]	[0..*] PowerSystemResources	PowerSystemResource	All power system resources at this location.
[0..1]	[0..*] ConfigurationEvents	ConfigurationEvent	All configuration events created for this location.
[0..*]	[0..1] CoordinateSystem	CoordinateSystem	Coordinate system used to describe position points of this location.
[1..1]	[0..*] PositionPoints	PositionPoint	Sequence of position points describing this location, expressed in coordinate system 'Location.CoordinateSystem'.
[0..1]	[0..*] Assets	Asset	All assets at this location.
[0..1]	[0..*] DiagramObjects	DiagramObject	inherited from: IdentifiedObject
[1..1]	[0..*] Names	Name	inherited from: IdentifiedObject

In the association end documentation (Table 3), the first column describes the multiplicity at this end of the association (i.e., how this class participates in association). The second column describes the other association end. Its multiplicity is included in brackets. The association end name is listed in plain text. The class at the other end of the association is in the third column. A multiplicity of zero indicates an optional association. A multiplicity of "*" indicates any number is allowed. For example, a multiplicity of [1..*] indicates a range from 1 to any larger number is allowed.

In the case that a class is an enumeration, the attributes table is replaced by the enums documentation as in Table 4, since enumeration literals have no the type. There are no inherited enumeration literals for an enumeration class.

Table 4 – Enums documentation

literal	description
steel	
lead	
lock	
other	

6 Top package IEC61968

6.1 General

Currently, normative parts of the model support the needs of information exchange defined in IEC 61968-3, IEC 61968-4, IEC 61968-9 and in IEC 61968-13.

Figure 29 shows package diagram IEC61968Dependencies.

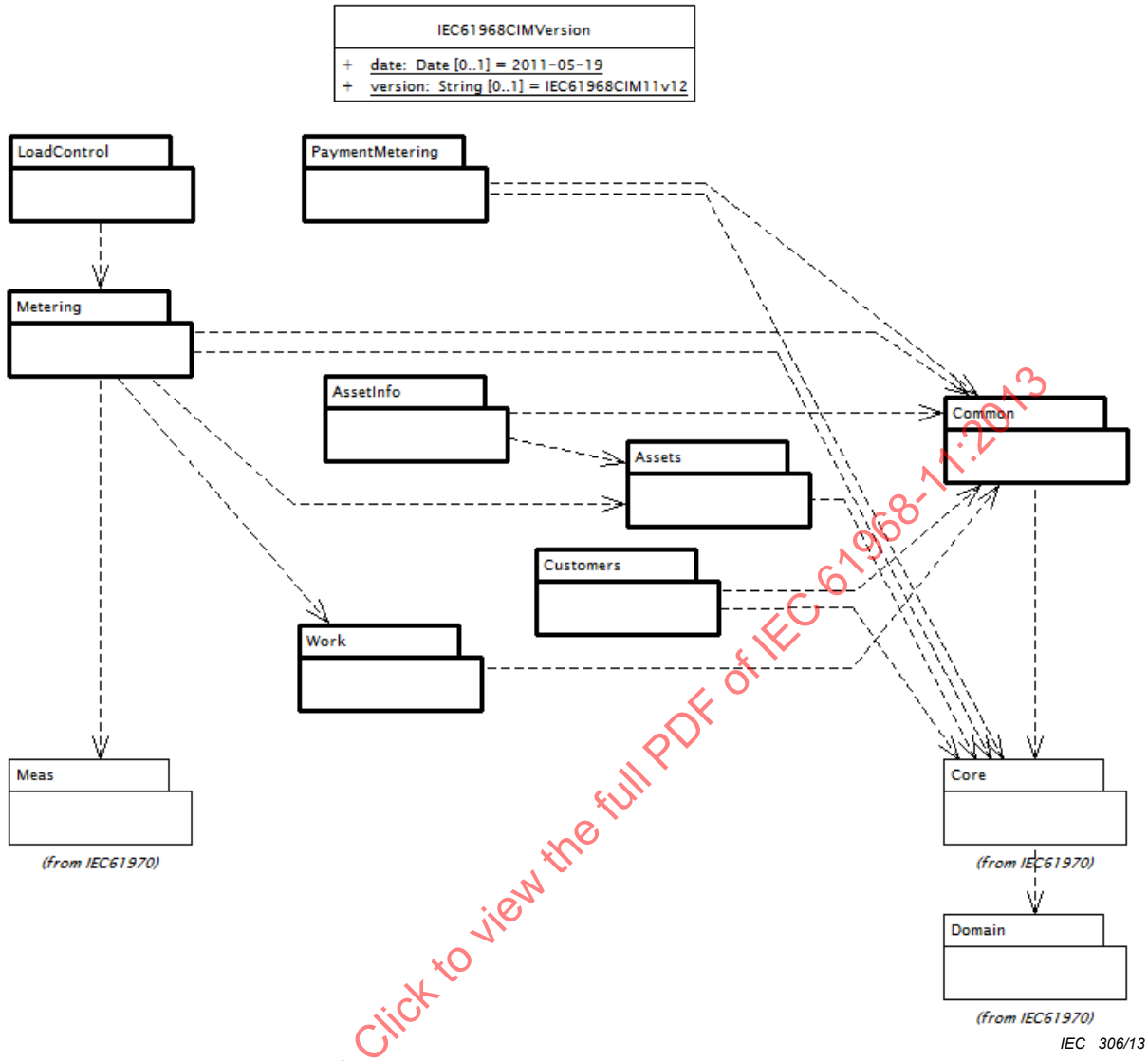


Figure 29 – Package diagram IEC61968::IEC61968Dependencies

This diagram shows version and normative contents of the CIM extensions for distribution as well as the dependencies among packages based on inheritance only.

Packages in bold are contained and documented in Clause 6.

6.2 IEC61968CIMVersion root class

IEC 61968 version number assigned to this UML model.

Table 5 shows all attributes of IEC61968CIMVersion.

Table 5 – Attributes of IEC61968::IEC61968CIMVersion

name	type	description
date=2011-08-10 (const)	Date	Form is YYYY-MM-DD for example for January 5, 2009 it is 2009-01-05.
version=IEC61968CIM11v13 (const)	String	Form is IEC61968CIMXXvYY where XX is the major CIM package version and the YY is the minor version. For example IEC61968CIM10v17.

6.3 Package Common

6.3.1 General

This package contains the information classes that support distribution management in general.

Figure 30 shows class diagram CommonInheritance.

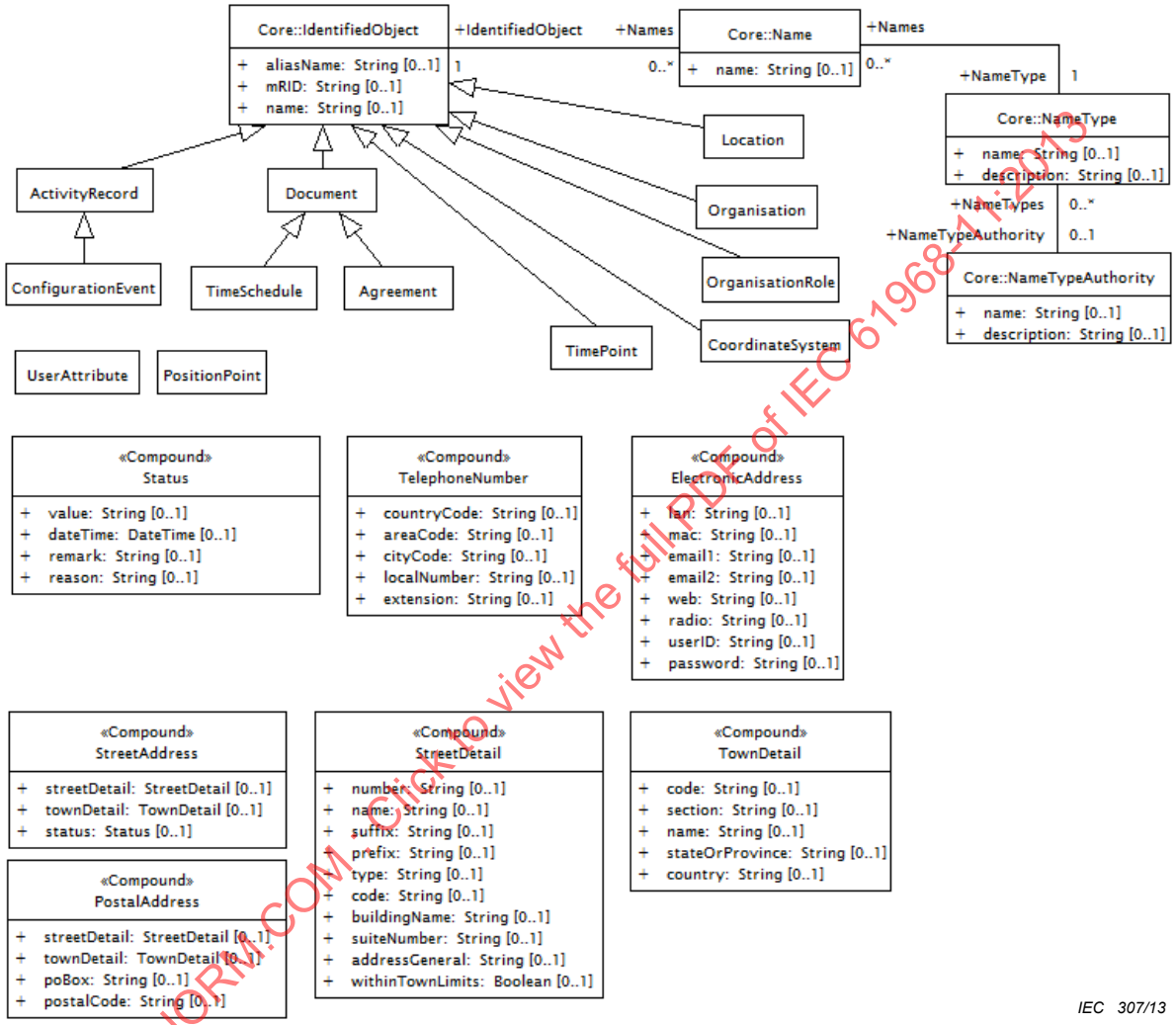


Figure 30 – Class diagram Common::CommonInheritance

This diagram shows inheritance hierarchy for normative classes from this package, as well as enumerations and compound types. It also shows the classes relevant to naming IdentifiedObject.

Figure 31 shows class diagram CommonOverview.

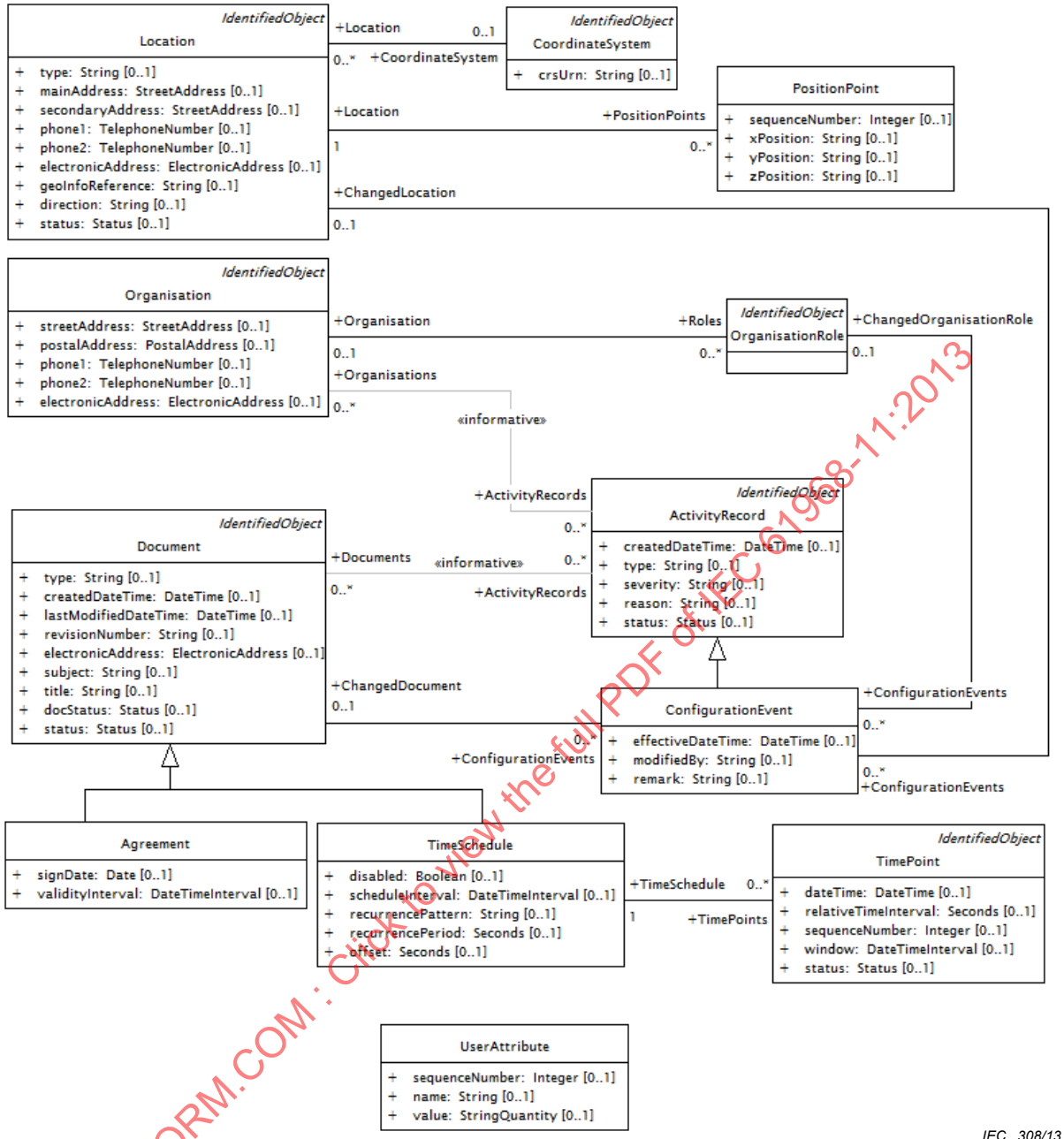


Figure 31 – Class diagram Common::CommonOverview

This diagram shows normative classes from this package.

6.3.2 Status compound

Current status information relevant to an entity.

Table 6 shows all attributes of Status.

Table 6 – Attributes of Common::Status

name	type	description
value	String	Status value at 'dateTime'; prior status changes may have been kept in instances of activity records associated with the object to which this status applies.
dateTime	DateTime	Date and time for which status 'value' applies.
remark	String	Pertinent information regarding the current 'value', as free form text.
reason	String	Reason code or explanation for why an object went to the current status 'value'.

6.3.3 PostalAddress compound

General purpose postal address information.

Table 7 shows all attributes of PostalAddress.

Table 7 – Attributes of Common::PostalAddress

name	type	description
streetDetail	StreetDetail	Street detail.
townDetail	TownDetail	Town detail.
poBox	String	Post office box.
postalCode	String	Postal code for the address.

6.3.4 StreetAddress compound

General purpose street address information.

Table 8 shows all attributes of StreetAddress.

Table 8 – Attributes of Common::StreetAddress

name	type	description
streetDetail	StreetDetail	Street detail.
townDetail	TownDetail	Town detail.
status	Status	Status of this address.

6.3.5 StreetDetail compound

Street details, in the context of address.

Table 9 shows all attributes of StreetDetail.

Table 9 – Attributes of Common::StreetDetail

name	type	description
number	String	Designator of the specific location on the street.
name	String	Name of the street.
suffix	String	Suffix to the street name. For example: North, South, East, West.

name	type	description
prefix	String	Prefix to the street name. For example: North, South, East, West.
type	String	Type of street. Examples include: street, circle, boulevard, avenue, road, drive, etc.
code	String	(if applicable) Utilities often make use of external reference systems, such as those of the town-planner's department or surveyor general's mapping system, that allocate global reference codes to streets.
buildingName	String	(if applicable) In certain cases the physical location of the place of interest does not have a direct point of entry from the street, but may be located inside a larger structure such as a building, complex, office block, apartment, etc.
suiteNumber	String	Number of the apartment or suite.
addressGeneral	String	Additional address information, for example a mailstop.
withinTownLimits	Boolean	True if this street is within the legal geographical boundaries of the specified town (default).

6.3.6 TownDetail compound

Town details, in the context of address.

Table 10 shows all attributes of TownDetail.

Table 10 – Attributes of Common::TownDetail

name	type	description
code	String	Town code.
section	String	Town section. For example, it is common for there to be 36 sections per township.
name	String	Town name.
stateOrProvince	String	Name of the state or province.
country	String	Name of the country.

6.3.7 ElectronicAddress compound

Electronic address information.

Table 11 shows all attributes of ElectronicAddress.

Table 11 – Attributes of Common::ElectronicAddress

name	type	description
lan	String	Address on local area network.
mac	String	MAC (Media Access Control) address.
email1	String	Primary email address.
email2	String	Alternate email address.
web	String	World wide web address.
radio	String	Radio address.
userID	String	User ID needed to log in, which can be for an individual person, an organisation, a location, etc.
password	String	Password needed to log in.

6.3.8 TelephoneNumber compound

Telephone number.

Table 12 shows all attributes of TelephoneNumber.

Table 12 – Attributes of Common::TelephoneNumber

name	type	description
countryCode	String	Country code.
areaCode	String	Area or region code.
cityCode	String	(if applicable) City code.
localNumber	String	Main (local) part of this telephone number.
extension	String	(if applicable) Extension for this telephone number.

6.3.9 ActivityRecord

Records activity for an entity at a point in time; activity may be for an event that has already occurred or for a planned activity.

Table 13 shows all attributes of ActivityRecord.

Table 13 – Attributes of Common::ActivityRecord

name	type	description
createdDateTime	DateTime	Date and time this activity record has been created (different from the 'status.dateTime', which is the time of a status change of the associated object, if applicable).
type	String	Type of event resulting in this activity record.
severity	String	Severity level of event resulting in this activity record.
reason	String	Reason for event resulting in this activity record, typically supplied when user initiated.
status	Status	Information on consequence of event resulting in this activity record.
aliasName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject

Table 14 shows all association ends of ActivityRecord with other classes.

Table 14 – Association ends of Common::ActivityRecord with other classes

[mult from]	[mult to] name	type	description
[0..*]	[0..*] Assets	Asset	All assets for which this activity record has been created.
[0..1]	[0..*] DiagramObjects	DiagramObject	inherited from: IdentifiedObject
[1..1]	[0..*] Names	Name	inherited from: IdentifiedObject

6.3.10 Agreement

Formal agreement between two parties defining the terms and conditions for a set of services. The specifics of the services are, in turn, defined via one or more service agreements.

Table 15 shows all attributes of Agreement.

Table 15 – Attributes of Common::Agreement

name	type	description
signDate	Date	Date this agreement was consummated among associated persons and/or organisations.
validityInterval	DateTimeInterval	Date and time interval this agreement is valid (from going into effect to termination).
type	String	inherited from: Document
createdDateTime	DateTime	inherited from: Document
lastModifiedDateTime	DateTime	inherited from: Document
revisionNumber	String	inherited from: Document
electronicAddress	ElectronicAddress	inherited from: Document
subject	String	inherited from: Document
title	String	inherited from: Document
docStatus	Status	inherited from: Document
status	Status	inherited from: Document
aliasName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject

Table 16 shows all association ends of Agreement with other classes.

Table 16 – Association ends of Common::Agreement with other classes

[mult from]	[mult to] name	type	description
[0..1]	[0..*] ConfigurationEvents	ConfigurationEvent	inherited from: Document
[0..1]	[0..*] DiagramObjects	DiagramObject	inherited from: IdentifiedObject
[1..1]	[0..*] Names	Name	inherited from: IdentifiedObject

6.3.11 ConfigurationEvent

Used to report details on creation, change or deletion of an entity or its configuration.

Table 17 shows all attributes of ConfigurationEvent.

Table 17 – Attributes of Common::ConfigurationEvent

name	type	description
effectiveDateTime	DateTime	Date and time this event has or will become effective.
modifiedBy	String	Source/initiator of modification.
remark	String	Free text remarks.
createdDateTime	DateTime	inherited from: ActivityRecord

name	type	description
type	String	inherited from: ActivityRecord
severity	String	inherited from: ActivityRecord
reason	String	inherited from: ActivityRecord
status	Status	inherited from: ActivityRecord
aliasName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject

Table 18 shows all association ends of ConfigurationEvent with other classes.

Table 18 – Association ends of Common::ConfigurationEvent with other classes

[mult from]	[mult to] name	type	description
[0..*]	[0..1] ChangedDocument	Document	Document whose change resulted in this configuration event.
[0..*]	[0..1] ChangedUsagePoint	UsagePoint	Usage point whose change resulted in this configuration event.
[0..*]	[0..1] ChangedAsset	Asset	Asset whose change resulted in this configuration event.
[0..*]	[0..1] ChangedServiceCategory	ServiceCategory	Service category whose change resulted in this configuration event.
[0..*]	[0..1] ChangedLocation	Location	Location whose change resulted in this configuration event.
[0..*]	[0..1] ChangedOrganisationRole	OrganisationRole	Organisation role whose change resulted in this configuration event.
[0..*]	[0..*] Assets	Asset	inherited from: ActivityRecord
[0..1]	[0..*] DiagramObjects	DiagramObject	inherited from: IdentifiedObject
[1..1]	[0..*] Names	Name	inherited from: IdentifiedObject

6.3.12 CoordinateSystem

Coordinate reference system.

Table 19 shows all attributes of CoordinateSystem.

Table 19 – Attributes of Common::CoordinateSystem

name	type	description
crsUrn	String	A Uniform Resource Name (URN) for the coordinate reference system (crs) used to define 'Location.PositionPoints'. An example would be the European Petroleum Survey Group (EPSG) code for a coordinate reference system, defined in URN under the Open Geospatial Consortium (OGC) namespace as: urn:ogc:def:uom:EPSG::XXXX, where XXXX is an EPSG code (a full list of codes can be found at the EPSG Registry website http://www.epsg-registry.org/). To define the coordinate system as being WGS84 (latitude, longitude) using an EPSG OGC, this attribute would be urn:ogc:def:uom:EPSG::4236. A profile should limit this code to a set of allowed URNs agreed to by all sending and receiving parties.
aliasName	String	inherited from: IdentifiedObject

name	type	description
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject

Table 20 shows all association ends of CoordinateSystem with other classes.

Table 20 – Association ends of Common::CoordinateSystem with other classes

[mult from]	[mult to] name	type	description
[0..1]	[0..*] Location	Location	All locations described with position points in this coordinate system.
[0..1]	[0..*] DiagramObjects	DiagramObject	inherited from: IdentifiedObject
[1..1]	[0..*] Names	Name	inherited from: IdentifiedObject

6.3.13 Document

Parent class for different groupings of information collected and managed as a part of a business process. It will frequently contain references to other objects, such as assets, people and power system resources.

Table 21 shows all attributes of Document.

Table 21 – Attributes of Common::Document

name	type	description
type	String	Utility-specific classification of this document, according to its corporate standards, practices, and existing IT systems (e.g., for management of assets, maintenance, work, outage, customers, etc.).
createdDateTime	DateTime	Date and time that this document was created.
lastModifiedDateTime	DateTime	Date and time this document was last modified. Documents may potentially be modified many times during their lifetime.
revisionNumber	String	Revision number for this document.
electronicAddress	ElectronicAddress	Electronic address.
subject	String	Document subject.
title	String	Document title.
docStatus	Status	Status of this document. For status of subject matter this document represents (e.g., Agreement, Work), use 'status' attribute. Example values for 'docStatus.status' are draft, approved, cancelled, etc.
status	Status	Status of subject matter (e.g., Agreement, Work) this document represents. For status of the document itself, use 'docStatus' attribute.
aliasName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject

Table 22 shows all association ends of Document with other classes.

Table 22 – Association ends of Common::Document with other classes

[mult from]	[mult to] name	type	description
[0..1]	[0..*] ConfigurationEvents	ConfigurationEvent	All configuration events created for this document.
[0..1]	[0..*] DiagramObjects	DiagramObject	inherited from: IdentifiedObject
[1..1]	[0..*] Names	Name	inherited from: IdentifiedObject

6.3.14 Location

The place, scene, or point of something where someone or something has been, is and/or will be at a given moment in time. It can be defined with one or more position points (coordinates) in a given coordinate system.

Table 23 shows all attributes of Location.

Table 23 – Attributes of Common::Location

name	type	description
type	String	Classification by utility's corporate standards and practices, relative to the location itself (e.g., geographical, functional accounting, etc., not a given property that happens to exist at that location).
mainAddress	StreetAddress	Main address of the location.
secondaryAddress	StreetAddress	Secondary address of the location. For example, PO Box address may have different ZIP code than that in the 'mainAddress'.
phone1	TelephoneNumber	Phone number.
phone2	TelephoneNumber	Additional phone number.
electronicAddress	ElectronicAddress	Electronic address.
geoInfoReference	String	(if applicable) Reference to geographical information source, often external to the utility.
direction	String	(if applicable) Direction that allows field crews to quickly find a given asset. For a given location, such as a street address, this is the relative direction in which to find the asset. For example, a streetlight may be located at the 'NW' (northwest) corner of the customer's site, or a usage point may be located on the second floor of an apartment building.
status	Status	Status of this location.
aliasName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject

Table 24 shows all association ends of Location with other classes.

Table 24 – Association ends of Common::Location with other classes

[mult from]	[mult to] name	type	description
[0..1]	[0..*] PowerSystemResources	PowerSystemResource	All power system resources at this location.
[0..1]	[0..*] ConfigurationEvents	ConfigurationEvent	All configuration events created for this location.
[0..*]	[0..1] CoordinateSystem	CoordinateSystem	Coordinate system used to describe

[mult from]	[mult to] name	type	description
			position points of this location.
[1..1]	[0..*] PositionPoints	PositionPoint	Sequence of position points describing this location, expressed in coordinate system 'Location.CoordinateSystem'.
[0..1]	[0..*] Assets	Asset	All assets at this location.
[0..1]	[0..*] DiagramObjects	DiagramObject	inherited from: IdentifiedObject
[1..1]	[0..*] Names	Name	inherited from: IdentifiedObject

6.3.15 Organisation

Organisation that might have roles as utility, contractor, supplier, manufacturer, customer, etc.

Table 25 shows all attributes of Organisation.

Table 25 – Attributes of Common::Organisation

name	type	description
streetAddress	StreetAddress	Street address.
postalAddress	PostalAddress	Postal address, potentially different than 'streetAddress' (e.g., another city).
phone1	TelephoneNumber	Phone number.
phone2	TelephoneNumber	Additional phone number.
electronicAddress	ElectronicAddress	Electronic address.
aliasName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject

Table 26 shows all association ends of Organisation with other classes.

Table 26 – Association ends of Common::Organisation with other classes

[mult from]	[mult to] name	type	description
[0..1]	[0..*] Roles	OrganisationRole	All roles of this organisation.
[0..1]	[0..*] DiagramObjects	DiagramObject	inherited from: IdentifiedObject
[1..1]	[0..*] Names	Name	inherited from: IdentifiedObject

6.3.16 OrganisationRole

Identifies a way in which an organisation may participate in the utility enterprise (e.g., customer, manufacturer, etc).

Table 27 shows all attributes of OrganisationRole.

Table 27 – Attributes of Common::OrganisationRole

name	type	description
aliasName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject

Table 28 shows all association ends of OrganisationRole with other classes.

Table 28 – Association ends of Common::OrganisationRole with other classes

[mult from]	[mult to] name	type	description
[0..1]	[0..*] ConfigurationEvents	ConfigurationEvent	All configuration events created for this organisation role.
[0..*]	[0..1] Organisation	Organisation	Organisation having this role.
[0..1]	[0..*] DiagramObjects	DiagramObject	inherited from: IdentifiedObject
[1..1]	[0..*] Names	Name	inherited from: IdentifiedObject

6.3.17 PositionPoint root class

Set of spatial coordinates that determine a point, defined in the coordinate system specified in 'Location.CoordinateSystem'. Use a single position point instance to describe a point-oriented location. Use a sequence of position points to describe a line-oriented object (physical location of non-point oriented objects like cables or lines), or area of an object (like a substation or a geographical zone – in this case, have first and last position point with the same values).

Table 29 shows all attributes of PositionPoint.

Table 29 – Attributes of Common::PositionPoint

name	type	description
sequenceNumber	Integer	Zero-relative sequence number of this point within a series of points.
xPosition	String	X axis position.
yPosition	String	Y axis position.
zPosition	String	(if applicable) Z axis position.

Table 30 shows all association ends of PositionPoint with other classes.

Table 30 – Association ends of Common::PositionPoint with other classes

[mult from]	[mult to] name	type	description
[0..*]	[1..1] Location	Location	Location described by this position point.

6.3.18 TimePoint

A point in time within a sequence of points in time relative to a TimeSchedule.

Table 31 shows all attributes of TimePoint.

Table 31 – Attributes of Common::TimePoint

name	type	description
dateTime	DateTime	Absolute date and time for this time point. For calendar-based time point, it is typically manually entered, while for interval-based or sequence-based time point it is derived.
relativeTimeInterval	Seconds	(if interval-based) A point in time relative to scheduled start time in 'TimeSchedule.scheduleInterval.start'.
sequenceNumber	Integer	(if sequence-based) Relative sequence number for this time point.
window	DateTimeInterval	Interval defining the window of time that this time point is valid (for example, seasonal, only on weekends, not on weekends, only 8:00 am to 5:00 am, etc.).
status	Status	Status of this time point.
aliasName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject

Table 32 shows all association ends of TimePoint with other classes.

Table 32 – Association ends of Common::TimePoint with other classes

[mult from]	[mult to] name	type	description
[0..*]	[1..1] TimeSchedule	TimeSchedule	Time schedule owning this time point.
[0..1]	[0..*] DiagramObjects	DiagramObject	inherited from: IdentifiedObject
[1..1]	[0..*] Names	Name	inherited from: IdentifiedObject

6.3.19 TimeSchedule

Description of anything that changes through time. Time schedule is used to perform a single-valued function of time. Use inherited 'type' attribute to give additional information on this schedule, such as: periodic (hourly, daily, weekly, monthly, etc.), day of the month, by date, calendar (specific times and dates).

Table 33 shows all attributes of TimeSchedule.

Table 33 – Attributes of Common::TimeSchedule

name	type	description
disabled	Boolean	True if this schedule is deactivated (disabled).
scheduleInterval	DateTimeInterval	Schedule date and time interval.
recurrencePattern	String	Interval at which the scheduled action repeats (e.g., first Monday of every month, last day of the month, etc.).
recurrencePeriod	Seconds	Duration between time points, from the beginning of one period to the beginning of the next period. Note that a device like a meter may have multiple interval periods (e.g., 1 min, 5 min, 15 min, 30 min, or 60 min).

name	type	description
offset	Seconds	The offset from midnight (i.e., 0 h, 0 min, 0 s) for the periodic time points to begin. For example, for an interval meter that is set up for five minute intervals ('recurrencePeriod'=300=5 min), setting 'offset'=120=2 min would result in scheduled events to read the meter executing at 2 min, 7 min, 12 min, 17 min, 22 min, 27 min, 32 min, 37 min, 42 min, 47 min, 52 min, and 57 min past each hour.
type	String	inherited from: Document
createdDateTime	DateTime	inherited from: Document
lastModifiedDateTime	DateTime	inherited from: Document
revisionNumber	String	inherited from: Document
electronicAddress	ElectronicAddress	inherited from: Document
subject	String	inherited from: Document
title	String	inherited from: Document
docStatus	Status	inherited from: Document
status	Status	inherited from: Document
aliasName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject

Table 34 shows all association ends of TimeSchedule with other classes.

Table 34 – Association ends of Common::TimeSchedule with other classes

[mult from]	[mult to] name	type	description
[1..1]	[0..*] TimePoints	TimePoint	Sequence of time points belonging to this time schedule.
[0..1]	[0..*] ConfigurationEvents	ConfigurationEvent	inherited from: Document
[0..1]	[0..*] DiagramObjects	DiagramObject	inherited from: IdentifiedObject
[1..1]	[0..*] Names	Name	inherited from: IdentifiedObject

6.3.20 UserAttribute root class

Generic name-value pair class, with optional sequence number and units for value; can be used to model parts of information exchange when concrete types are not known in advance.

Table 35 shows all attributes of UserAttribute.

Table 35 – Attributes of Common::UserAttribute

name	type	description
sequenceNumber	Integer	Sequence number for this attribute in a list of attributes.
name	String	Name of an attribute.
value	StringQuantity	Value of an attribute, including unit information.

Table 36 shows all association ends of UserAttribute with other classes.

Table 36 – Association ends of Common::UserAttribute with other classes

[mult from]	[mult to] name	type	description
[0..*]	[0..1] Transaction	Transaction	Transaction for which this snapshot has been recorded.

6.4 Package Assets

6.4.1 General

This package contains the core information classes that support asset management applications that deal with the physical and lifecycle aspects of various network resources (as opposed to power system resource models defined in IEC61970::Wires package, which support network applications).

Figure 32 shows class diagram AssetsInheritance.

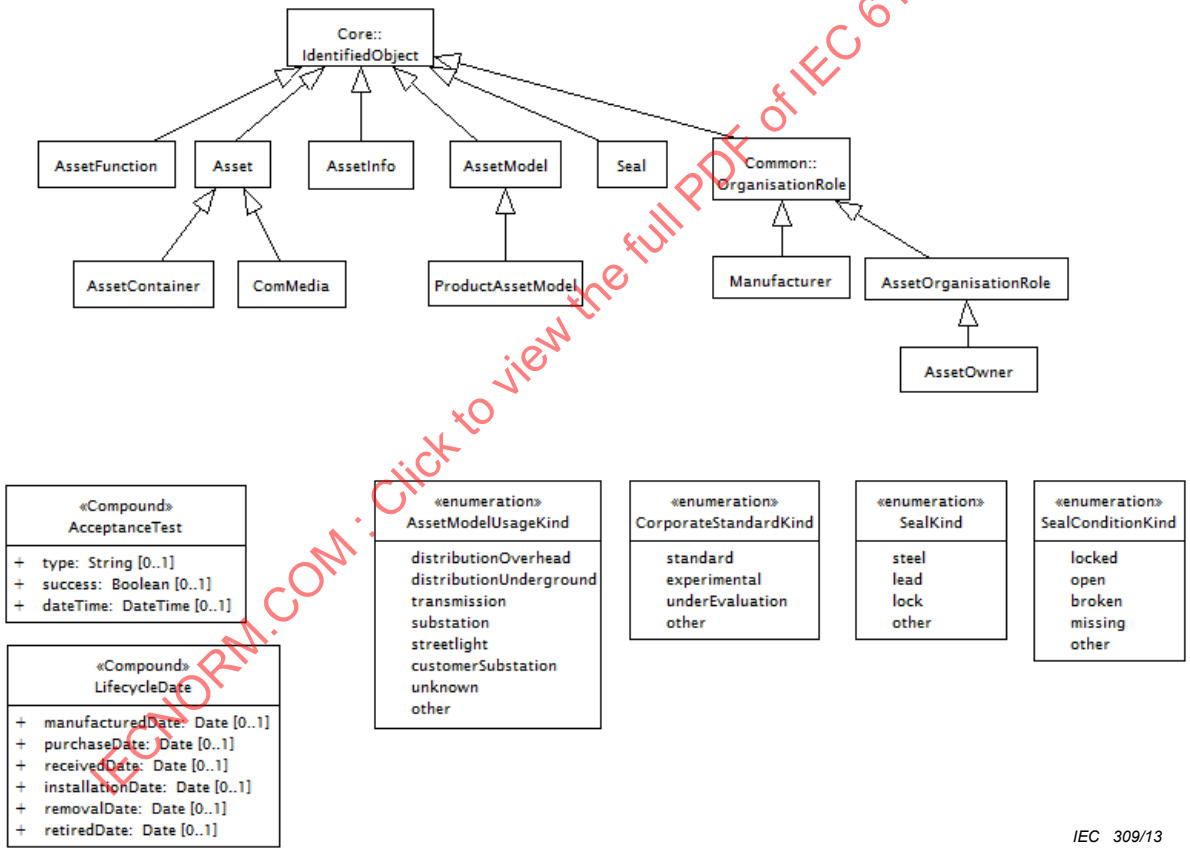


Figure 32 – Class diagram Assets::AssetsInheritance

This diagram shows the inheritance hierarchy for normative classes from this package, as well as enumerations and compound types.

Figure 33 shows class diagram AssetsOverview.

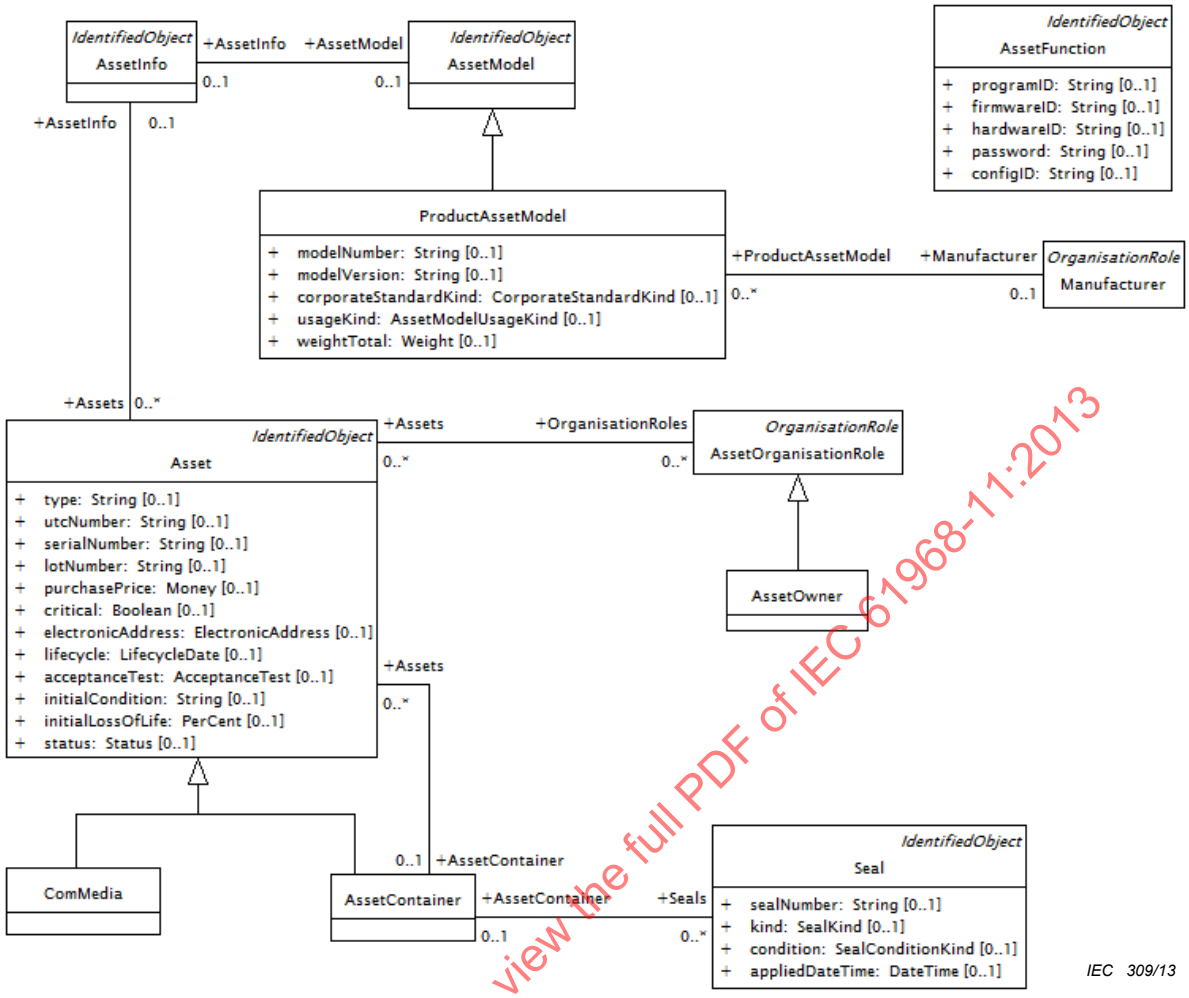


Figure 33 – Class diagram Assets::AssetsOverview

This diagram shows normative classes from this package.

6.4.2 AcceptanceTest compound

Acceptance test for assets.

Table 37 shows all attributes of AcceptanceTest.

Table 37 – Attributes of Assets::AcceptanceTest

name	type	description
type	String	Type of test or group of tests that was conducted on 'dateTime'.
success	Boolean	True if asset has passed acceptance test and may be placed in or is in service. It is set to false if asset is removed from service and is required to be tested again before being placed back in service, possibly in a new location. Since asset may go through multiple tests during its lifecycle, the date of each acceptance test may be recorded in Asset.ActivityRecord.status.dateTime.
dateTime	DateTime	Date and time the asset was last tested using the 'type' of test and yielding the current status in 'success' attribute.

6.4.3 LifecycleDate compound

Dates for lifecycle events of an asset.

Table 38 shows all attributes of LifecycleDate.

Table 38 – Attributes of Assets::LifecycleDate

name	type	description
manufacturedDate	Date	Date the asset was manufactured.
purchaseDate	Date	Date the asset was purchased. Note that even though an asset may have been purchased, it may not have been received into inventory at the time of purchase.
receivedDate	Date	Date the asset was received and first placed into inventory.
installationDate	Date	(if applicable) Date current installation was completed, which may not be the same as the in-service date. Asset may have been installed at other locations previously. Ignored if asset is (1) not currently installed (e.g., stored in a depot) or (2) not intended to be installed (e.g., vehicle, tool).
removalDate	Date	(if applicable) Date when the asset was last removed from service. Ignored if (1) not intended to be in service, or (2) currently in service.
retiredDate	Date	(if applicable) Date the asset is permanently retired from service and may be scheduled for disposal. Ignored if asset is (1) currently in service, or (2) permanently removed from service.

6.4.4 AssetModelUsageKind enumeration

Usage for an asset model.

Table 39 shows all literals of AssetModelUsageKind.

Table 39 – Literals of Assets::AssetModelUsageKind

literal	description
distributionOverhead	Asset model is intended for use in distribution overhead network.
distributionUnderground	Asset model is intended for use in underground distribution network.
transmission	Asset model is intended for use in transmission network.
substation	Asset model is intended for use in substation.
streetlight	Asset model is intended for use as streetlight.
customerSubstation	Asset model is intended for use in customer substation.
unknown	Usage of the asset model is unknown.
other	Other kind of asset model usage.

6.4.5 CorporateStandardKind enumeration

Kind of corporate standard.

Table 40 shows all literals of CorporateStandardKind.

Table 40 – Literals of Assets::CorporateStandardKind

literal	description
standard	Asset model is used as corporate standard.
experimental	Asset model is used experimentally.
underEvaluation	Asset model usage is under evaluation.
other	Other kind of corporate standard for the asset model.

6.4.6 SealConditionKind enumeration

Kind of seal condition.

Table 41 shows all literals of SealConditionKind.

Table 41 – Literals of Assets::SealConditionKind

literal	description
locked	Seal is locked.
open	Seal is open.
broken	Seal is broken.
missing	Seal is missing.
other	Other kind of seal condition.

6.4.7 SealKind enumeration

Kind of seal.

Table 42 shows all literals of SealKind.

Table 42 – Literals of Assets::SealKind

literal	description
steel	Steel seal.
lead	Lead seal.
lock	Lock seal.
other	Other kind of seal.

6.4.8 Asset

Tangible resource of the utility, including power system equipment, various end devices, cabinets, buildings, etc. For electrical network equipment, the role of the asset is defined through PowerSystemResource and its subclasses, defined mainly in the Wires model (refer to IEC61970-301 and model package IEC61970::Wires). Asset description places emphasis on the physical characteristics of the equipment fulfilling that role.

Table 43 shows all attributes of Asset.

Table 43 – Attributes of Assets::Asset

name	type	description
type	String	Utility-specific classification of Asset and its subtypes, according to their corporate standards, practices, and existing IT systems (e.g., for management of assets, maintenance, work, outage, customers, etc.).
utcNumber	String	Uniquely tracked commodity (UTC) number.
serialNumber	String	Serial number of this asset.
lotNumber	String	Lot number for this asset. Even for the same model and version number, many assets are manufactured in lots.
purchasePrice	Money	Purchase price of asset.
critical	Boolean	True if asset is considered critical for some reason (for example, a pole with critical attachments).
electronicAddress	ElectronicAddress	Electronic address.
lifecycle	LifecycleDate	Lifecycle dates for this asset.
acceptanceTest	AcceptanceTest	Information on acceptance test.
initialCondition	String	Condition of asset in inventory or at time of installation. Examples include new, rebuilt, overhaul required, other. Refer to inspection data for information on the most current condition of the asset.
initialLossOfLife	PerCent	Whenever an asset is reconditioned, percentage of expected life for the asset when it was new, zero for new devices.
status	Status	Status of this asset.
aliasName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject

Table 44 shows all association ends of Asset with other classes.

Table 44 – Association ends of Assets::Asset with other classes

[mult from]	[mult to] name	type	description
[0..*]	[0..*] PowerSystemResources	PowerSystemResource	All power system resources used to electrically model this asset. For example, transformer asset is electrically modelled with a transformer and its windings and tap changer.
[0..*]	[0..*] ActivityRecords	ActivityRecord	All activity records created for this asset.
[0..1]	[0..*] ConfigurationEvents	ConfigurationEvent	All configuration events created for this asset.
[0..*]	[0..1] Location	Location	Location of this asset.
[0..*]	[0..*] OrganisationRoles	AssetOrganisationRole	All roles an organisation plays for this asset.
[0..*]	[0..1] AssetContainer	AssetContainer	Container of this asset.
[0..*]	[0..1] AssetInfo	AssetInfo	Data applicable to this asset.
[0..1]	[0..*] DiagramObjects	DiagramObject	inherited from: IdentifiedObject
[1..1]	[0..*] Names	Name	inherited from: IdentifiedObject

6.4.9 AssetContainer

Asset that is aggregation of other assets such as conductors, transformers, switchgear, land, fences, buildings, equipment, vehicles, etc.

Table 45 shows all attributes of AssetContainer.

Table 45 – Attributes of Assets::AssetContainer

name	type	description
type	String	inherited from: Asset
utcNumber	String	inherited from: Asset
serialNumber	String	inherited from: Asset
lotNumber	String	inherited from: Asset
purchasePrice	Money	inherited from: Asset
critical	Boolean	inherited from: Asset
electronicAddress	ElectronicAddress	inherited from: Asset
lifecycle	LifecycleDate	inherited from: Asset
acceptanceTest	AcceptanceTest	inherited from: Asset
initialCondition	String	inherited from: Asset
initialLossOfLife	PerCent	inherited from: Asset
status	Status	inherited from: Asset
aliasName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject

Table 46 shows all association ends of AssetContainer with other classes.

Table 46 – Association ends of Assets::AssetContainer with other classes

[mult from]	[mult to] name	type	description
[0..1]	[0..*] Seals	Seal	All seals applied to this asset container.
[0..1]	[0..*] Assets	Asset	All assets within this container asset.
[0..*]	[0..*] PowerSystemResources	PowerSystemResource	inherited from: Asset
[0..*]	[0..*] ActivityRecords	ActivityRecord	inherited from: Asset
[0..1]	[0..*] ConfigurationEvents	ConfigurationEvent	inherited from: Asset
[0..*]	[0..1] Location	Location	inherited from: Asset
[0..*]	[0..*] OrganisationRoles	AssetOrganisationRole	inherited from: Asset
[0..*]	[0..1] AssetContainer	AssetContainer	inherited from: Asset
[0..*]	[0..1] AssetInfo	AssetInfo	inherited from: Asset
[0..1]	[0..*] DiagramObjects	DiagramObject	inherited from: IdentifiedObject
[1..1]	[0..*] Names	Name	inherited from: IdentifiedObject

6.4.10 AssetFunction

Function performed by an asset.

Table 47 shows all attributes of AssetFunction.

Table 47 – Attributes of Assets::AssetFunction

name	type	description
programID	String	Name of program.
firmwareID	String	Firmware version.
hardwareID	String	Hardware version.
password	String	Password needed to access this function.
configID	String	Configuration specified for this function.
aliasName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject

Table 48 shows all association ends of AssetFunction with other classes.

Table 48 – Association ends of Assets::AssetFunction with other classes

[mult from]	[mult to] name	type	description
[0..1]	[0..*] DiagramObjects	DiagramObject	inherited from: IdentifiedObject
[1..1]	[0..*] Names	Name	inherited from: IdentifiedObject

6.4.11 AssetInfo

Set of attributes of an asset, representing typical datasheet information of a physical device that can be instantiated and shared in different data exchange contexts:

– as attributes of an asset instance (installed or in stock)
– as attributes of an asset model (product by a manufacturer)
– as attributes of a type asset (generic type of an asset as used in designs/extension planning).

Table 49 shows all attributes of AssetInfo.

Table 49 – Attributes of Assets::AssetInfo

name	type	description
aliasName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject

Table 50 shows all association ends of AssetInfo with other classes.

Table 50 – Association ends of Assets::AssetInfo with other classes

[mult from]	[mult to] name	type	description
[0..1]	[0..*] PowerSystemResource	PowerSystemResource	All power system resources with this datasheet information.
[0..1]	[0..*] Assets	Asset	All assets described by this data.
[0..1]	[0..1] AssetModel	AssetModel	Asset model described by this data.
[0..1]	[0..*] DiagramObjects	DiagramObject	inherited from: IdentifiedObject
[1..1]	[0..*] Names	Name	inherited from: IdentifiedObject

6.4.12 AssetModel

Model of an asset, either a product of a specific manufacturer or a generic asset model or material item. Datasheet characteristics are available through the associated AssetInfo subclass and can be shared with asset or power system resource instances.

Table 51 shows all attributes of AssetModel.

Table 51 – Attributes of Assets::AssetModel

name	type	description
aliasName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject

Table 52 shows all association ends of AssetModel with other classes.

Table 52 – Association ends of Assets::AssetModel with other classes

[mult from]	[mult to] name	type	description
[0..1]	[0..1] AssetInfo	AssetInfo	Data applicable to this asset model.
[0..1]	[0..*] DiagramObjects	DiagramObject	inherited from: IdentifiedObject
[1..1]	[0..*] Names	Name	inherited from: IdentifiedObject

6.4.13 AssetOrganisationRole

Role an organisation plays with respect to asset.

Table 53 shows all attributes of AssetOrganisationRole.

Table 53 – Attributes of Assets::AssetOrganisationRole

name	type	description
aliasName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject

Table 54 shows all association ends of AssetOrganisationRole with other classes.

Table 54 – Association ends of Assets::AssetOrganisationRole with other classes

[mult from]	[mult to] name	type	description
[0..*]	[0..*] Assets	Asset	All assets for this organisation role.
[0..1]	[0..*] ConfigurationEvents	ConfigurationEvent	inherited from: OrganisationRole
[0..*]	[0..1] Organisation	Organisation	inherited from: OrganisationRole
[0..1]	[0..*] DiagramObjects	DiagramObject	inherited from: IdentifiedObject
[1..1]	[0..*] Names	Name	inherited from: IdentifiedObject

6.4.14 AssetOwner

Owner of the asset.

Table 55 shows all attributes of AssetOwner.

Table 55 – Attributes of Assets::AssetOwner

name	type	description
aliasName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject

Table 56 shows all association ends of AssetOwner with other classes.

Table 56 – Association ends of Assets::AssetOwner with other classes

[mult from]	[mult to] name	type	description
[0..*]	[0..*] Assets	Asset	inherited from: AssetOrganisationRole
[0..1]	[0..*] ConfigurationEvents	ConfigurationEvent	inherited from: OrganisationRole
[0..*]	[0..1] Organisation	Organisation	inherited from: OrganisationRole
[0..1]	[0..*] DiagramObjects	DiagramObject	inherited from: IdentifiedObject
[1..1]	[0..*] Names	Name	inherited from: IdentifiedObject

6.4.15 ComMedia

Communication media such as fibre optic cable, power-line, telephone, etc.

Table 57 shows all attributes of ComMedia.

Table 57 – Attributes of Assets::ComMedia

name	type	description
type	String	inherited from: Asset
utcNumber	String	inherited from: Asset
serialNumber	String	inherited from: Asset
lotNumber	String	inherited from: Asset
purchasePrice	Money	inherited from: Asset

name	type	description
critical	Boolean	inherited from: Asset
electronicAddress	ElectronicAddress	inherited from: Asset
lifecycle	LifecycleDate	inherited from: Asset
acceptanceTest	AcceptanceTest	inherited from: Asset
initialCondition	String	inherited from: Asset
initialLossOfLife	PerCent	inherited from: Asset
status	Status	inherited from: Asset
aliasName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject

Table 58 shows all association ends of ComMedia with other classes.

Table 58 – Association ends of Assets::ComMedia with other classes

[mult from]	[mult to] name	type	description
[0..*]	[0..*] PowerSystemResources	PowerSystemResource	inherited from: Asset
[0..*]	[0..*] ActivityRecords	ActivityRecord	inherited from: Asset
[0..1]	[0..*] ConfigurationEvents	ConfigurationEvent	inherited from: Asset
[0..*]	[0..1] Location	Location	inherited from: Asset
[0..*]	[0..*] OrganisationRoles	AssetOrganisationRole	inherited from: Asset
[0..*]	[0..1] AssetContainer	AssetContainer	inherited from: Asset
[0..*]	[0..1] AssetInfo	AssetInfo	inherited from: Asset
[0..1]	[0..*] DiagramObjects	DiagramObject	inherited from: IdentifiedObject
[1..1]	[0..*] Names	Name	inherited from: IdentifiedObject

6.4.16 Manufacturer

Organisation that manufactures asset products.

Table 59 shows all attributes of Manufacturer.

Table 59 – Attributes of Assets::Manufacturer

name	type	description
aliasName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject

Table 60 shows all association ends of Manufacturer with other classes.

Table 60 – Association ends of Assets::Manufacturer with other classes

[mult from]	[mult to] name	type	description
[0..1]	[0..*] ProductAssetModel	ProductAssetModel	All asset models by this manufacturer.
[0..1]	[0..*] ConfigurationEvents	ConfigurationEvent	inherited from: OrganisationRole
[0..*]	[0..1] Organisation	Organisation	inherited from: OrganisationRole
[0..1]	[0..*] DiagramObjects	DiagramObject	inherited from: IdentifiedObject
[1..1]	[0..*] Names	Name	inherited from: IdentifiedObject

6.4.17 ProductAssetModel

Asset model by a specific manufacturer.

Table 61 shows all attributes of ProductAssetModel.

Table 61 – Attributes of Assets::ProductAssetModel

name	type	description
modelNumber	String	Manufacturer's model number.
modelVersion	String	Version number for product model, which indicates vintage of the product.
corporateStandardKind	CorporateStandardKind	Kind of corporate standard for this asset model.
usageKind	AssetModelUsageKind	Intended usage for this asset model.
weightTotal	Weight	Total manufactured weight of asset.
aliasName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject

Table 62 shows all association ends of ProductAssetModel with other classes.

Table 62 – Association ends of Assets::ProductAssetModel with other classes

[mult from]	[mult to] name	type	description
[0..*]	[0..1] Manufacturer	Manufacturer	Manufacturer of this asset model.
[0..1]	[0..1] AssetInfo	AssetInfo	inherited from: AssetModel
[0..1]	[0..*] DiagramObjects	DiagramObject	inherited from: IdentifiedObject
[1..1]	[0..*] Names	Name	inherited from: IdentifiedObject

6.4.18 Seal

Physically controls access to AssetContainers.

Table 63 shows all attributes of Seal.

Table 63 – Attributes of Assets::Seal

name	type	description
sealNumber	String	(reserved word) Seal number.
kind	SealKind	Kind of seal.
condition	SealConditionKind	Condition of seal.
appliedDateTime	DateTime	Date and time this seal has been applied.
aliasName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject

Table 64 shows all association ends of Seal with other classes.

Table 64 – Association ends of Assets::Seal with other classes

[mult from]	[mult to] name	type	description
[0..*]	[0..1] AssetContainer	AssetContainer	Asset container to which this seal is applied.
[0..1]	[0..*] DiagramObjects	DiagramObject	inherited from: IdentifiedObject
[1..1]	[0..*] Names	Name	inherited from: IdentifiedObject

6.5 Package AssetInfo

6.5.1 General

This package is an extension of Assets package and contains the core information classes that support asset management and different network and work planning applications with specialized AssetInfo subclasses. They hold attributes that can be referenced by not only Asset-s or AssetModel-s but also by ConductingEquipment-s.

Figure 34 shows class diagram AssetInfoInheritance.

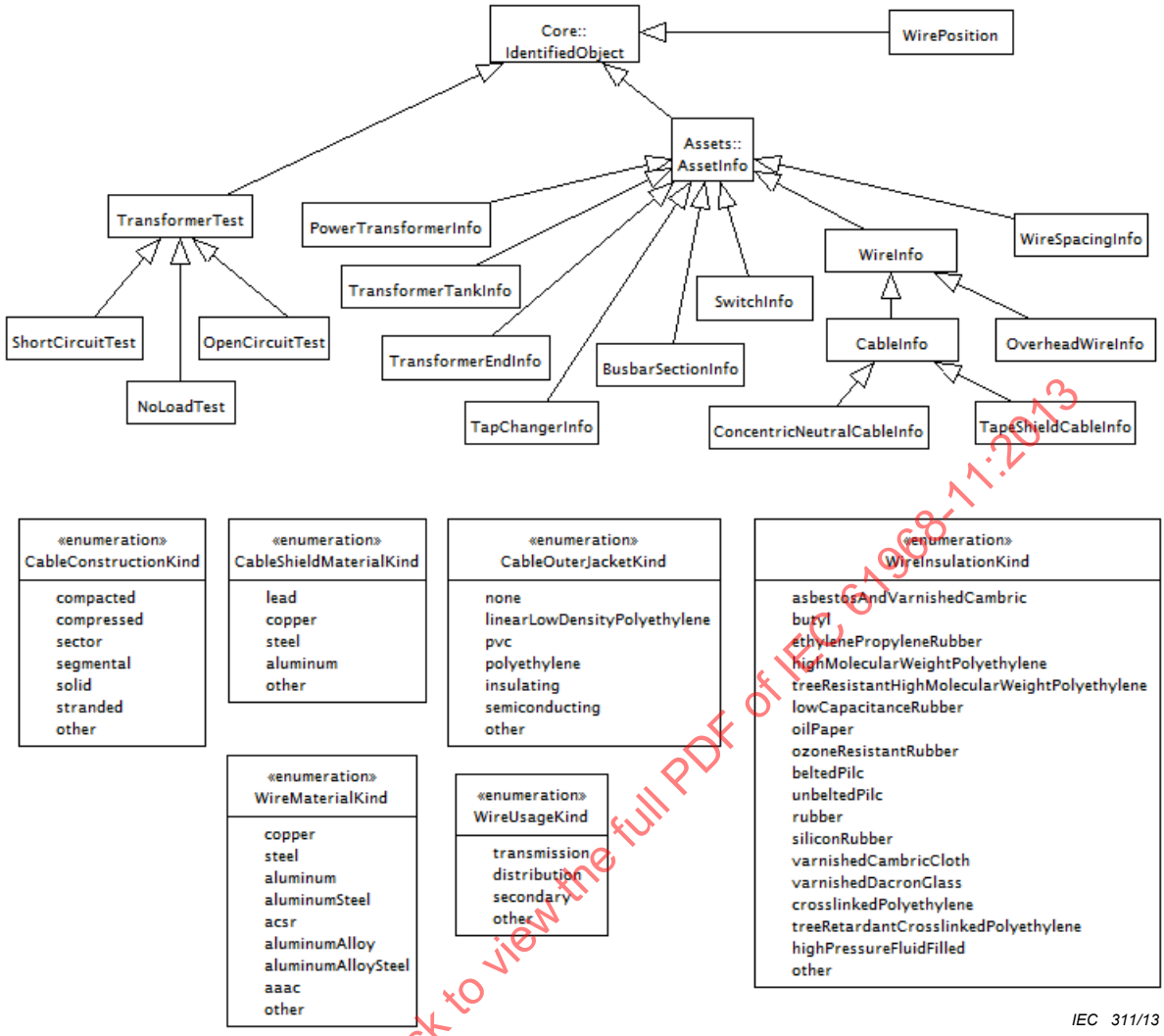


Figure 34 – Class diagram AssetInfo::AssetInfoInheritance

This diagram shows the inheritance hierarchy for normative classes from this package, as well as enumerations and compound types.

Figure 35 shows class diagram AssetInfoOverview.

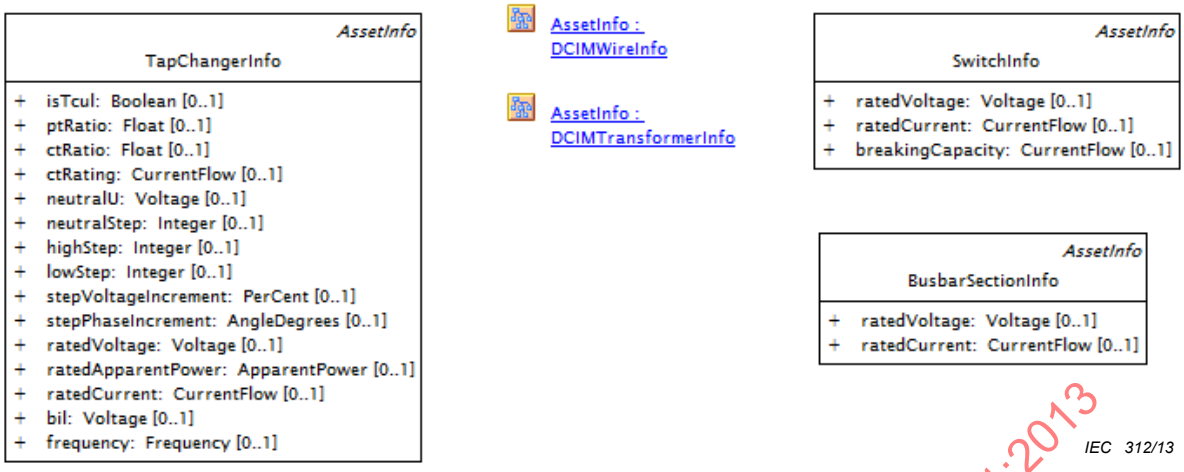


Figure 35 – Class diagram AssetInfo::AssetInfoOverview

This diagram shows normative classes from this package.

Figure 36 shows class diagram DCIMWireInfo.

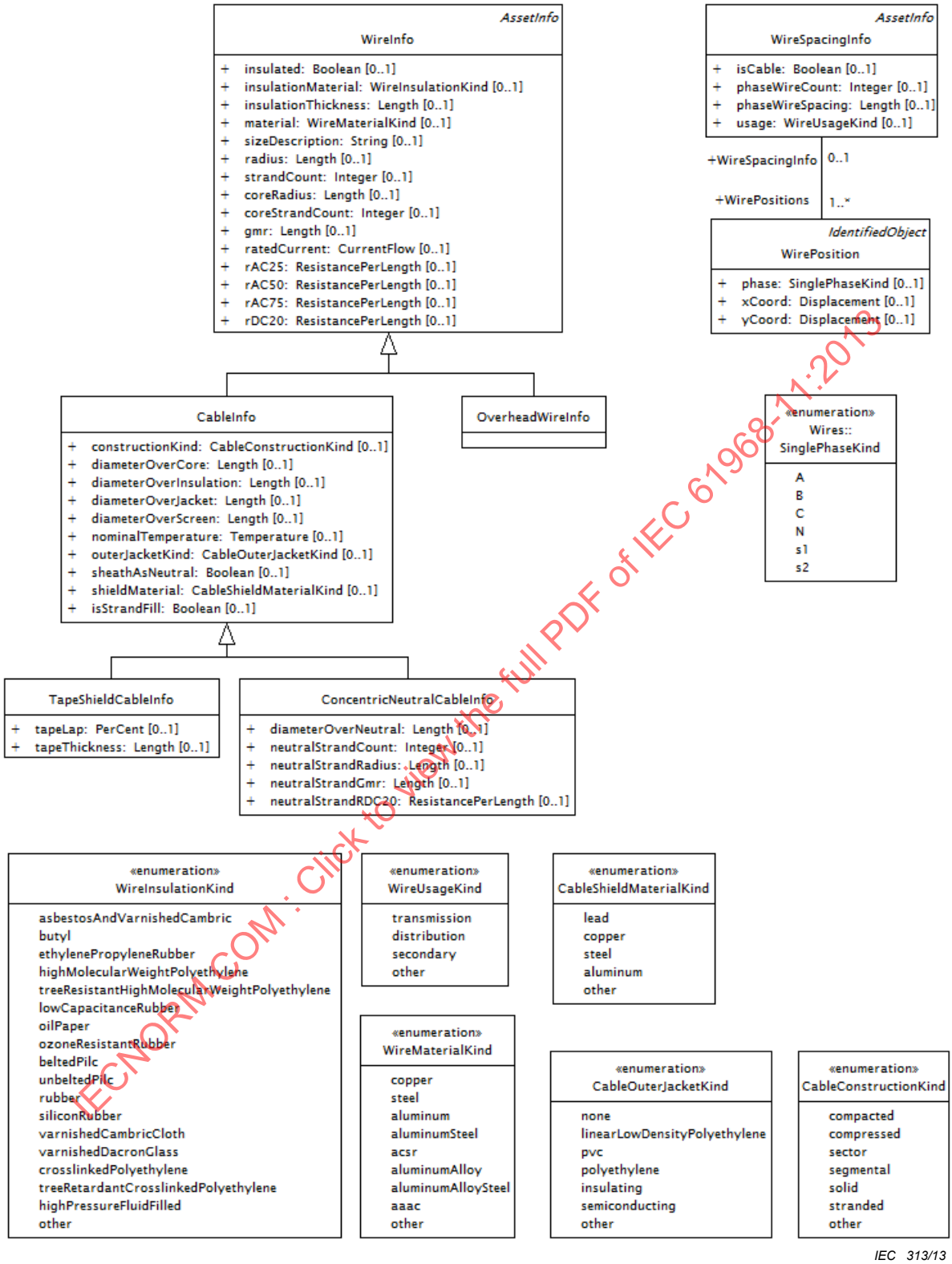


Figure 36 – Class diagram AssetInfo::DCIMWireInfo

This diagram shows classes used to model physical aspects of lines in DCIM.

Figure 37 shows class diagram DCIMTransformerInfo.

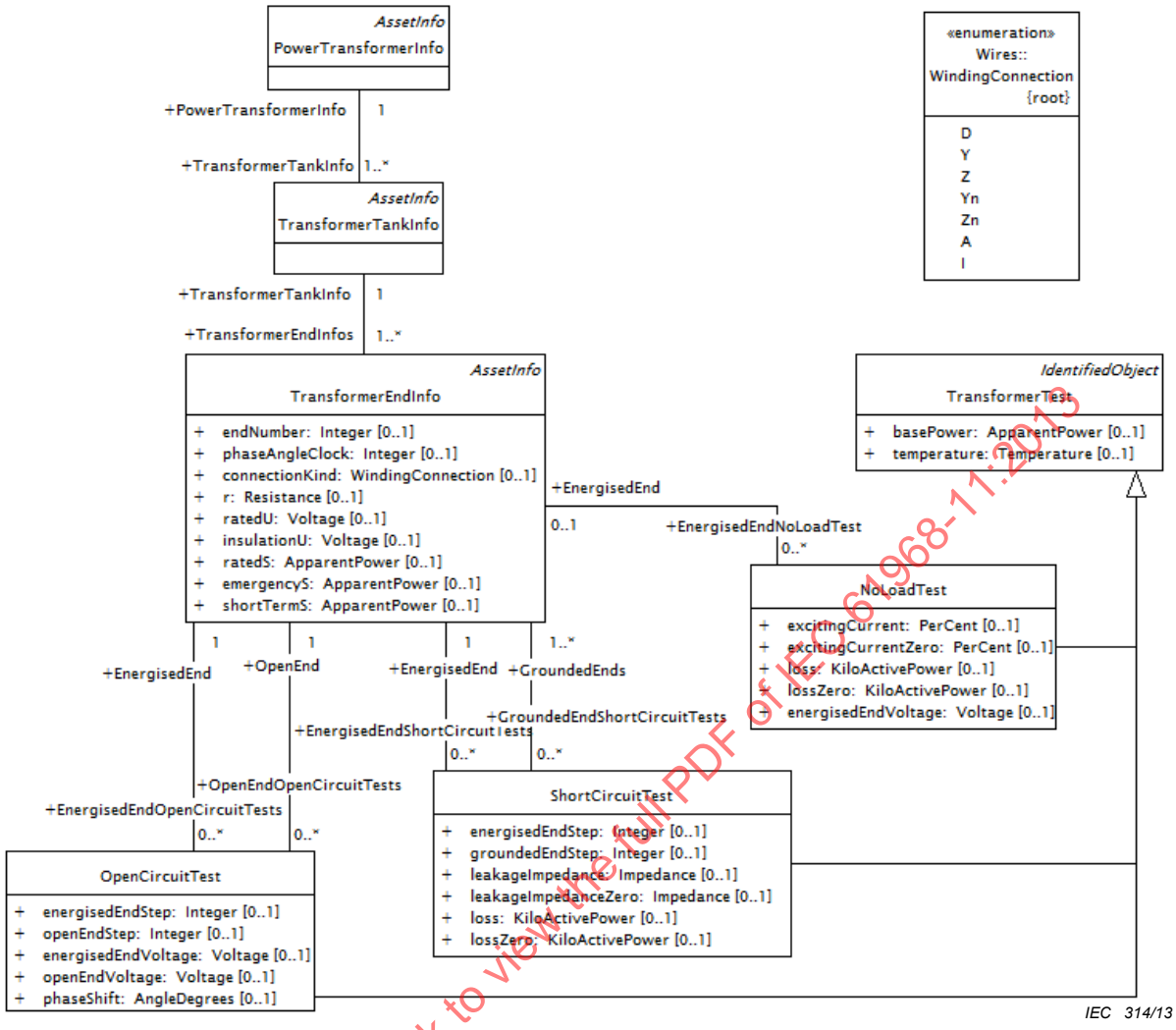


Figure 37 – Class diagram AssetInfo::DCIMTransformerInfo

This diagram shows one part of classes used to model transformers in DCIM.

6.5.2 BusbarSectionInfo

Busbar section data.

Table 65 shows all attributes of BusbarSectionInfo.

Table 65 – Attributes of AssetInfo::BusbarSectionInfo

name	type	description
ratedVoltage	Voltage	Rated voltage.
ratedCurrent	CurrentFlow	Rated current.
aliasName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject

Table 66 shows all association ends of BusbarSectionInfo with other classes.

Table 66 – Association ends of AssetInfo::BusbarSectionInfo with other classes

[mult from]	[mult to] name	type	description
[0..1]	[0..*] PowerSystemResource	PowerSystemResource	inherited from: AssetInfo
[0..1]	[0..*] Assets	Asset	inherited from: AssetInfo
[0..1]	[0..1] AssetModel	AssetModel	inherited from: AssetInfo
[0..1]	[0..*] DiagramObjects	DiagramObject	inherited from: IdentifiedObject
[1..1]	[0..*] Names	Name	inherited from: IdentifiedObject

6.5.3 CableConstructionKind enumeration

Kind of cable construction.

Table 67 shows all literals of CableConstructionKind.

Table 67 – Literals of AssetInfo::CableConstructionKind

literal	description
compacted	Compacted cable.
compressed	Compressed cable.
sector	Sector cable.
segmental	Segmental cable.
solid	Solid cable.
stranded	Stranded cable.
other	Other kind of cable construction.

6.5.4 CableInfo

Cable data.

Table 68 shows all attributes of CableInfo.

Table 68 – Attributes of AssetInfo::CableInfo

name	type	description
constructionKind	CableConstructionKind	Kind of construction of this cable.
diameterOverCore	Length	Diameter over the core, including any semi-con screen; should be the insulating layer's inside diameter.
diameterOverInsulation	Length	Diameter over the insulating layer, excluding outer screen.
diameterOverJacket	Length	Diameter over the outermost jacketing layer.
diameterOverScreen	Length	Diameter over the outer screen; should be the shield's inside diameter.
nominalTemperature	Temperature	Maximum nominal design operating temperature.
outerJacketKind	CableOuterJacketKind	Kind of outer jacket of this cable.
sheathAsNeutral	Boolean	True if sheath / shield is used as a neutral (i.e., bonded).
shieldMaterial	CableShieldMaterialKind	Material of the shield.
isStrandFill	Boolean	True if wire strands are extruded in a way to fill the voids in the cable.

name	type	description
insulated	Boolean	inherited from: WireInfo
insulationMaterial	WireInsulationKind	inherited from: WireInfo
insulationThickness	Length	inherited from: WireInfo
material	WireMaterialKind	inherited from: WireInfo
sizeDescription	String	inherited from: WireInfo
radius	Length	inherited from: WireInfo
strandCount	Integer	inherited from: WireInfo
coreRadius	Length	inherited from: WireInfo
coreStrandCount	Integer	inherited from: WireInfo
gmr	Length	inherited from: WireInfo
ratedCurrent	CurrentFlow	inherited from: WireInfo
rAC25	ResistancePerLength	inherited from: WireInfo
rAC50	ResistancePerLength	inherited from: WireInfo
rAC75	ResistancePerLength	inherited from: WireInfo
rDC20	ResistancePerLength	inherited from: WireInfo
aliasName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject

Table 69 shows all association ends of CableInfo with other classes.

Table 69 – Association ends of AssetInfo::CableInfo with other classes

[mult from]	[mult to] name	type	description
[0..*]	[0..*] Impedances	PerLengthImpedance	inherited from: WireInfo
[0..1]	[0..*] PowerSystemResource	PowerSystemResource	inherited from: AssetInfo
[0..1]	[0..*] Assets	Asset	inherited from: AssetInfo
[0..1]	[0..1] AssetModel	AssetModel	inherited from: AssetInfo
[0..1]	[0..*] DiagramObjects	DiagramObject	inherited from: IdentifiedObject
[1..1]	[0..*] Names	Name	inherited from: IdentifiedObject

6.5.5 CableOuterJacketKind enumeration

Kind of cable outer jacket.

Table 70 shows all literals of CableOuterJacketKind.

Table 70 – Literals of AssetInfo::CableOuterJacketKind

literal	description
none	Cable has no outer jacket.
linearLowDensityPolyethylene	Linear low density polyethylene cable outer jacket.
pvc	PVC cable outer jacket.
polyethylene	Polyethylene cable outer jacket.

literal	description
insulating	Insulating cable outer jacket.
semiconducting	Semiconducting cable outer jacket.
other	Pther kind of cable outer jacket.

6.5.6 CableShieldMaterialKind enumeration

Kind of cable shield material.

Table 71 shows all literals of CableShieldMaterialKind.

Table 71 – Literals of AssetInfo::CableShieldMaterialKind

literal	description
lead	Lead cable shield.
copper	Copper cable shield.
steel	Steel cable shield.
aluminum	Aluminum cable shield.
other	Other kind of cable shield material.

6.5.7 ConcentricNeutralCableInfo

Concentric neutral cable data.

Table 72 shows all attributes of ConcentricNeutralCableInfo.

Table 72 – Attributes of AssetInfo::ConcentricNeutralCableInfo

name	type	description
diameterOverNeutral	Length	Diameter over the concentric neutral strands.
neutralStrandCount	Integer	Number of concentric neutral strands.
neutralStrandRadius	Length	Outside radius of the neutral strand.
neutralStrandGmr	Length	Geometric mean radius of the neutral strand.
neutralStrandRDC20	ResistancePerLength	DC resistance per unit length of the neutral strand at 20 °C.
constructionKind	CableConstructionKind	inherited from: CableInfo
diameterOverCore	Length	inherited from: CableInfo
diameterOverInsulation	Length	inherited from: CableInfo
diameterOverJacket	Length	inherited from: CableInfo
diameterOverScreen	Length	inherited from: CableInfo
nominalTemperature	Temperature	inherited from: CableInfo
outerJacketKind	CableOuterJacketKind	inherited from: CableInfo
sheathAsNeutral	Boolean	inherited from: CableInfo
shieldMaterial	CableShieldMaterialKind	inherited from: CableInfo
isStrandFill	Boolean	inherited from: CableInfo
insulated	Boolean	inherited from: WireInfo
insulationMaterial	WireInsulationKind	inherited from: WireInfo
insulationThickness	Length	inherited from: WireInfo
material	WireMaterialKind	inherited from: WireInfo

name	type	description
sizeDescription	String	inherited from: WireInfo
radius	Length	inherited from: WireInfo
strandCount	Integer	inherited from: WireInfo
coreRadius	Length	inherited from: WireInfo
coreStrandCount	Integer	inherited from: WireInfo
gmr	Length	inherited from: WireInfo
ratedCurrent	CurrentFlow	inherited from: WireInfo
rAC25	ResistancePerLength	inherited from: WireInfo
rAC50	ResistancePerLength	inherited from: WireInfo
rAC75	ResistancePerLength	inherited from: WireInfo
rDC20	ResistancePerLength	inherited from: WireInfo
aliasName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject

Table 73 shows all association ends of ConcentricNeutralCableInfo with other classes.

Table 73 – Association ends of AssetInfo::ConcentricNeutralCableInfo with other classes

[mult from]	[mult to] name	type	description
[0..*]	[0..*] Impedances	PerLengthImpedance	inherited from: WireInfo
[0..1]	[0..*] PowerSystemResource	PowerSystemResource	inherited from: AssetInfo
[0..1]	[0..*] Assets	Asset	inherited from: AssetInfo
[0..1]	[0..1] AssetModel	AssetModel	inherited from: AssetInfo
[0..1]	[0..*] DiagramObjects	DiagramObject	inherited from: IdentifiedObject
[1..1]	[0..*] Names	Name	inherited from: IdentifiedObject

6.5.8 NoLoadTest

No-load test results determine core admittance parameters. They include exciting current and core loss measurements from applying voltage to one winding. The excitation may be positive sequence or zero sequence. The test may be repeated at different voltages to measure saturation.

Table 74 shows all attributes of NoLoadTest.

Table 74 – Attributes of AssetInfo::NoLoadTest

name	type	description
excitingCurrent	PerCent	Exciting current measured from a positive-sequence or single-phase excitation test.
excitingCurrentZero	PerCent	Exciting current measured from a zero-sequence open-circuit excitation test.
loss	KiloActivePower	Losses measured from a positive-sequence or single-phase excitation test.
lossZero	KiloActivePower	Losses measured from a zero-sequence excitation test.

name	type	description
energisedEndVoltage	Voltage	Voltage applied to the winding (end) during test.
basePower	ApparentPower	inherited from: TransformerTest
temperature	Temperature	inherited from: TransformerTest
aliasName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject

Table 75 shows all association ends of NoLoadTest with other classes.

Table 75 – Association ends of AssetInfo::NoLoadTest with other classes

[mult from]	[mult to] name	type	description
[0..*]	[0..1] EnergisedEnd	TransformerEndInfo	Transformer end that current is applied to in this no-load test.
[0..1]	[0..*] DiagramObjects	DiagramObject	inherited from: IdentifiedObject
[1..1]	[0..*] Names	Name	inherited from: IdentifiedObject

6.5.9 OpenCircuitTest

Open-circuit test results verify winding turn ratios and phase shifts. They include induced voltage and phase shift measurements on open-circuit windings, with voltage applied to the energised end. For three-phase windings, the excitation can be a positive sequence (the default) or a zero sequence.

Table 76 shows all attributes of OpenCircuitTest.

Table 76 – Attributes of AssetInfo::OpenCircuitTest

name	type	description
energisedEndStep	Integer	Tap step number for the energised end of the test pair.
openEndStep	Integer	Tap step number for the open end of the test pair.
energisedEndVoltage	Voltage	Voltage applied to the winding (end) during test.
openEndVoltage	Voltage	Voltage measured at the open-circuited end, with the energised end set to rated voltage and all other ends open.
phaseShift	AngleDegrees	Phase shift measured at the open end with the energised end set to rated voltage and all other ends open.
basePower	ApparentPower	inherited from: TransformerTest
temperature	Temperature	inherited from: TransformerTest
aliasName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject

Table 77 shows all association ends of OpenCircuitTest with other classes.

Table 77 – Association ends of AssetInfo::OpenCircuitTest with other classes

[mult from]	[mult to] name	type	description
[0..*]	[1..1] OpenEnd	TransformerEndInfo	Transformer end measured for induced voltage and angle in this open-circuit test.
[0..*]	[1..1] EnergisedEnd	TransformerEndInfo	Transformer end that current is applied to in this open-circuit test.
[0..1]	[0..*] DiagramObjects	DiagramObject	inherited from: IdentifiedObject
[1..1]	[0..*] Names	Name	inherited from: IdentifiedObject

6.5.10 OverheadWireInfo

Overhead wire data.

Table 78 shows all attributes of OverheadWireInfo.

Table 78 – Attributes of AssetInfo::OverheadWireInfo

name	type	description
insulated	Boolean	inherited from: WireInfo
insulationMaterial	WireInsulationKind	inherited from: WireInfo
insulationThickness	Length	inherited from: WireInfo
material	WireMaterialKind	inherited from: WireInfo
sizeDescription	String	inherited from: WireInfo
radius	Length	inherited from: WireInfo
strandCount	Integer	inherited from: WireInfo
coreRadius	Length	inherited from: WireInfo
coreStrandCount	Integer	inherited from: WireInfo
gmr	Length	inherited from: WireInfo
ratedCurrent	CurrentFlow	inherited from: WireInfo
rAC25	ResistancePerLength	inherited from: WireInfo
rAC50	ResistancePerLength	inherited from: WireInfo
rAC75	ResistancePerLength	inherited from: WireInfo
rDC20	ResistancePerLength	inherited from: WireInfo
aliasName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject

Table 79 shows all association ends of OverheadWireInfo with other classes.

Table 79 – Association ends of AssetInfo::OverheadWireInfo with other classes

[mult from]	[mult to] name	type	description
[0..*]	[0..*] Impedances	PerLengthImpedance	inherited from: WireInfo
[0..1]	[0..*] PowerSystemResource	PowerSystemResource	inherited from: AssetInfo
[0..1]	[0..*] Assets	Asset	inherited from: AssetInfo
[0..1]	[0..1] AssetModel	AssetModel	inherited from: AssetInfo
[0..1]	[0..*] DiagramObjects	DiagramObject	inherited from: IdentifiedObject
[1..1]	[0..*] Names	Name	inherited from: IdentifiedObject

6.5.11 PowerTransformerInfo

Set of power transformer data, from an equipment library.

Table 80 shows all attributes of PowerTransformerInfo.

Table 80 – Attributes of AssetInfo::PowerTransformerInfo

name	type	description
aliasName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject

Table 81 shows all association ends of PowerTransformerInfo with other classes.

Table 81 – Association ends of AssetInfo::PowerTransformerInfo with other classes

[mult from]	[mult to] name	type	description
[1..1]	[1..*] TransformerTankInfo	TransformerTankInfo	Data for all the tanks described by this power transformer data.
[0..1]	[0..*] PowerSystemResource	PowerSystemResource	inherited from: AssetInfo
[0..1]	[0..*] Assets	Asset	inherited from: AssetInfo
[0..1]	[0..1] AssetModel	AssetModel	inherited from: AssetInfo
[0..1]	[0..*] DiagramObjects	DiagramObject	inherited from: IdentifiedObject
[1..1]	[0..*] Names	Name	inherited from: IdentifiedObject

6.5.12 ShortCircuitTest

Short-circuit test results determine mesh impedance parameters. They include load losses and leakage impedances. For three-phase windings, the excitation can be a positive sequence (the default) or a zero sequence. There shall be at least one grounded winding.

Table 82 shows all attributes of ShortCircuitTest.

Table 82 – Attributes of AssetInfo::ShortCircuitTest

name	type	description
energisedEndStep	Integer	Tap step number for the energised end of the test pair.
groundedEndStep	Integer	Tap step number for the grounded end of the test pair.
leakageImpedance	Impedance	Leakage impedance measured from a positive-sequence or single-phase short-circuit test.
leakageImpedanceZero	Impedance	Leakage impedance measured from a zero-sequence short-circuit test.
loss	KiloActivePower	Load losses from a positive-sequence or single-phase short-circuit test.
lossZero	KiloActivePower	Load losses from a zero-sequence short-circuit test.
basePower	ApparentPower	inherited from: TransformerTest
temperature	Temperature	inherited from: TransformerTest
aliasName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject

Table 83 shows all association ends of ShortCircuitTest with other classes.

Table 83 – Association ends of AssetInfo::ShortCircuitTest with other classes

[mult from]	[mult to] name	type	description
[0..*]	[1..*] GroundedEnds	TransformerEndInfo	All ends short-circuited in this short-circuit test.
[0..*]	[1..1] EnergisedEnd	TransformerEndInfo	Transformer end that voltage is applied to in this short-circuit test. The test voltage is chosen to induce rated current in the energised end.
[0..1]	[0..*] DiagramObjects	DiagramObject	inherited from: IdentifiedObject
[1..1]	[0..*] Names	Name	inherited from: IdentifiedObject

6.5.13 SwitchInfo

Switch data.

Table 84 shows all attributes of SwitchInfo.

Table 84 – Attributes of AssetInfo::SwitchInfo

name	type	description
ratedVoltage	Voltage	Rated voltage.
ratedCurrent	CurrentFlow	Rated current.
breakingCapacity	CurrentFlow	The maximum fault current a breaking device can break safely under prescribed conditions of use.
aliasName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject

Table 85 shows all association ends of SwitchInfo with other classes.

Table 85 – Association ends of AssetInfo::SwitchInfo with other classes

[mult from]	[mult to] name	type	description
[0..1]	[0..*] PowerSystemResource	PowerSystemResource	inherited from: AssetInfo
[0..1]	[0..*] Assets	Asset	inherited from: AssetInfo
[0..1]	[0..1] AssetModel	AssetModel	inherited from: AssetInfo
[0..1]	[0..*] DiagramObjects	DiagramObject	inherited from: IdentifiedObject
[1..1]	[0..*] Names	Name	inherited from: IdentifiedObject

6.5.14 TapChangerInfo

Tap changer data.

Table 86 shows all attributes of TapChangerInfo.

Table 86 – Attributes of AssetInfo::TapChangerInfo

name	type	description
isTcul	Boolean	Whether this tap changer has under load tap changing capabilities.
ptRatio	Float	Built-in voltage transducer ratio.
ctRatio	Float	Built-in current transducer ratio.
ctRating	CurrentFlow	Built-in current transformer primary rating.
neutralU	Voltage	Voltage at which the winding operates at the neutral tap setting.
neutralStep	Integer	The neutral tap step position for the winding.
highStep	Integer	Highest possible tap step position, advance from neutral.
lowStep	Integer	Lowest possible tap step position, retard from neutral.
stepVoltageIncrement	PerCent	Tap step increment, in per cent of rated voltage, per step position.
stepPhaseIncrement	AngleDegrees	Phase shift per step position.
ratedVoltage	Voltage	Rated voltage.
ratedApparentPower	ApparentPower	Rated apparent power.
ratedCurrent	CurrentFlow	Rated current.
bil	Voltage	Basic Insulation Level (BIL) expressed as the impulse crest voltage of a nominal wave, typically 1,2 X 50 µs. This is a measure of the ability of the insulation to withstand very high voltage surges.
frequency	Frequency	Frequency at which the ratings apply.
aliasName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject

Table 87 shows all association ends of TapChangerInfo with other classes.

Table 87 – Association ends of AssetInfo::TapChangerInfo with other classes

[mult from]	[mult to] name	type	description
[0..1]	[0..*] PowerSystemResource	PowerSystemResource	inherited from: AssetInfo
[0..1]	[0..*] Assets	Asset	inherited from: AssetInfo

[mult from]	[mult to] name	type	description
[0..1]	[0..1] AssetModel	AssetModel	inherited from: AssetInfo
[0..1]	[0..*] DiagramObjects	DiagramObject	inherited from: IdentifiedObject
[1..1]	[0..*] Names	Name	inherited from: IdentifiedObject

6.5.15 TapeShieldCableInfo

Tape shield cable data.

Table 88 shows all attributes of TapeShieldCableInfo.

Table 88 – Attributes of AssetInfo::TapeShieldCableInfo

name	type	description
tapeLap	PerCent	Percentage of the tape shield width that overlaps in each wrap, typically 10 % to 25 %.
tapeThickness	Length	Thickness of the tape shield, before wrapping.
constructionKind	CableConstructionKind	inherited from: CableInfo
diameterOverCore	Length	inherited from: CableInfo
diameterOverInsulation	Length	inherited from: CableInfo
diameterOverJacket	Length	inherited from: CableInfo
diameterOverScreen	Length	inherited from: CableInfo
nominalTemperature	Temperature	inherited from: CableInfo
outerJacketKind	CableOuterJacketKind	inherited from: CableInfo
sheathAsNeutral	Boolean	inherited from: CableInfo
shieldMaterial	CableShieldMaterialKind	inherited from: CableInfo
isStrandFill	Boolean	inherited from: CableInfo
insulated	Boolean	inherited from: WireInfo
insulationMaterial	WireInsulationKind	inherited from: WireInfo
insulationThickness	Length	inherited from: WireInfo
material	WireMaterialKind	inherited from: WireInfo
sizeDescription	String	inherited from: WireInfo
radius	Length	inherited from: WireInfo
strandCount	Integer	inherited from: WireInfo
coreRadius	Length	inherited from: WireInfo
coreStrandCount	Integer	inherited from: WireInfo
gmr	Length	inherited from: WireInfo
ratedCurrent	CurrentFlow	inherited from: WireInfo
rAC25	ResistancePerLength	inherited from: WireInfo
rAC50	ResistancePerLength	inherited from: WireInfo
rAC75	ResistancePerLength	inherited from: WireInfo
rDC20	ResistancePerLength	inherited from: WireInfo
aliasName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject

Table 89 shows all association ends of TapeShieldCableInfo with other classes.

Table 89 – Association ends of AssetInfo::TapeShieldCableInfo with other classes

[mult from]	[mult to] name	type	description
[0..*]	[0..*] Impedances	PerLengthImpedance	inherited from: WireInfo
[0..1]	[0..*] PowerSystemResource	PowerSystemResource	inherited from: AssetInfo
[0..1]	[0..*] Assets	Asset	inherited from: AssetInfo
[0..1]	[0..1] AssetModel	AssetModel	inherited from: AssetInfo
[0..1]	[0..*] DiagramObjects	DiagramObject	inherited from: IdentifiedObject
[1..1]	[0..*] Names	Name	inherited from: IdentifiedObject

6.5.16 TransformerEndInfo

Transformer end data.

Table 90 shows all attributes of TransformerEndInfo.

Table 90 – Attributes of AssetInfo::TransformerEndInfo

name	type	description
endNumber	Integer	Number for this transformer end, corresponding to the end's order in the PowerTransformer.vectorGroup attribute. Highest voltage winding should be 1.
phaseAngleClock	Integer	Winding phase angle where 360 degrees are represented with clock hours, so the valid values are {0, ..., 11}. For example, to express the second winding in code 'Dyn11', set attributes as follows: 'endNumber'=2, 'connectionKind' = Yn and 'phaseAngleClock' = 11.
connectionKind	WindingConnection	Kind of connection.
r	Resistance	DC resistance.
ratedU	Voltage	Rated voltage: phase-phase for three-phase windings, and either phase-phase or phase-neutral for single-phase windings.
insulationU	Voltage	Basic insulation level voltage rating.
ratedS	ApparentPower	Normal apparent power rating.
emergencyS	ApparentPower	Apparent power that the winding can carry under emergency conditions (also called long-term emergency power).
shortTermS	ApparentPower	Apparent power that this winding can carry for a short period of time (in emergency).
aliasName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject

Table 91 shows all association ends of TransformerEndInfo with other classes.

Table 91 – Association ends of AssetInfo::TransformerEndInfo with other classes

[mult from]	[mult to] name	type	description
[0..1]	[0..1] CoreAdmittance	TransformerCoreAdmittance	Core admittance calculated from this transformer end datasheet, representing magnetising current and core losses. The full values of the transformer should be supplied for one transformer end info only.
[0..1]	[0..*] FromMeshImpedance	TransformerMeshImpedance	All mesh impedances between this 'to' and other 'from' transformer ends.
[0..*]	[0..*] ToMeshImpedances	TransformerMeshImpedance	All mesh impedances between this 'from' and other 'to' transformer ends.
[0..1]	[0..1] TransformerStarImpedance	TransformerStarImpedance	Transformer star impedance calculated from this transformer end datasheet.
[0..1]	[0..*] EnergisedEndNoLoadTest	NoLoadTest	All no-load test measurements in which this transformer end was energised.
[1..1]	[0..*] OpenEndOpenCircuitTests	OpenCircuitTest	All open-circuit test measurements in which this transformer end was not excited.
[1..1]	[0..*] EnergisedEndOpenCircuitTests	OpenCircuitTest	All open-circuit test measurements in which this transformer end was excited.
[1..*]	[0..*] GroundedEndShortCircuitTests	ShortCircuitTest	All short-circuit test measurements in which this transformer end was short-circuited.
[1..1]	[0..*] EnergisedEndShortCircuitTests	ShortCircuitTest	All short-circuit test measurements in which this transformer end was energised.
[1..*]	[1..1] TransformerTankInfo	TransformerTankInfo	Transformer tank data that this end description is part of.
[0..1]	[0..*] PowerSystemResource	PowerSystemResource	inherited from: AssetInfo
[0..1]	[0..*] Assets	Asset	inherited from: AssetInfo
[0..1]	[0..1] AssetModel	AssetModel	inherited from: AssetInfo
[0..1]	[0..*] DiagramObjects	DiagramObject	inherited from: IdentifiedObject
[1..1]	[0..*] Names	Name	inherited from: IdentifiedObject

6.5.17 TransformerTankInfo

Set of transformer tank data, from an equipment library.

Table 92 shows all attributes of TransformerTankInfo.

Table 92 – Attributes of AssetInfo::TransformerTankInfo

name	type	description
aliasName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject

Table 93 shows all association ends of TransformerTankInfo with other classes.

Table 93 – Association ends of AssetInfo::TransformerTankInfo with other classes

[mult from]	[mult to] name	type	description
[1..*]	[1..1] PowerTransformerInfo	PowerTransformerInfo	Power transformer data that this tank description is part of.
[1..1]	[1..*] TransformerEndInfos	TransformerEndInfo	Data for all the ends described by this transformer tank data.
[0..1]	[0..*] PowerSystemResource	PowerSystemResource	inherited from: AssetInfo
[0..1]	[0..*] Assets	Asset	inherited from: AssetInfo
[0..1]	[0..1] AssetModel	AssetModel	inherited from: AssetInfo
[0..1]	[0..*] DiagramObjects	DiagramObject	inherited from: IdentifiedObject
[1..1]	[0..*] Names	Name	inherited from: IdentifiedObject

6.5.18 TransformerTest

Test result for transformer ends, such as short-circuit, open-circuit (excitation) or no-load test.

Table 94 shows all attributes of TransformerTest.

Table 94 – Attributes of AssetInfo::TransformerTest

name	type	description
basePower	ApparentPower	Base power at which the tests are conducted, usually equal to the rateds of one of the involved transformer ends.
temperature	Temperature	Temperature at which the test is conducted.
aliasName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject

Table 95 shows all association ends of TransformerTest with other classes.

Table 95 – Association ends of AssetInfo::TransformerTest with other classes

[mult from]	[mult to] name	type	description
[0..1]	[0..*] DiagramObjects	DiagramObject	inherited from: IdentifiedObject
[1..1]	[0..*] Names	Name	inherited from: IdentifiedObject

6.5.19 WireInfo

Wire data that can be specified per line segment phase, or for the line segment as a whole in case its phases all have the same wire characteristics.

Table 96 shows all attributes of WireInfo.

Table 96 – Attributes of AssetInfo::WireInfo

name	type	description
insulated	Boolean	True if conductor is insulated.
insulationMaterial	WireInsulationKind	(if insulated conductor) Material used for insulation.
insulationThickness	Length	(if insulated conductor) Thickness of the insulation.
material	WireMaterialKind	Conductor material.
sizeDescription	String	Describes the wire gauge or cross section (e.g., 4/0, #2, 336.5).
radius	Length	Outside radius of the wire.
strandCount	Integer	Number of strands in the conductor.
coreRadius	Length	(if there is a different core material) Radius of the central core.
coreStrandCount	Integer	(if used) Number of strands in the steel core.
gmr	Length	Geometric mean radius. If we replace the conductor by a thin walled tube of radius GMR, then its reactance is identical to the reactance of the actual conductor.
ratedCurrent	CurrentFlow	Current carrying capacity of the wire under stated thermal conditions.
rAC25	ResistancePerLength	AC resistance per unit length of the conductor at 25 °C.
rAC50	ResistancePerLength	AC resistance per unit length of the conductor at 50 °C.
rAC75	ResistancePerLength	AC resistance per unit length of the conductor at 75 °C.
rDC20	ResistancePerLength	DC resistance per unit length of the conductor at 20 °C.
aliasName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject

Table 97 shows all association ends of WireInfo with other classes.

Table 97 – Association ends of AssetInfo::WireInfo with other classes

[mult from]	[mult to] name	type	description
[0..*]	[0..*] Impedances	PerLengthImpedance	All impedances calculated from this wire datasheet.
[0..1]	[0..*] PowerSystemResource	PowerSystemResource	inherited from: AssetInfo
[0..1]	[0..*] Assets	Asset	inherited from: AssetInfo
[0..1]	[0..1] AssetModel	AssetModel	inherited from: AssetInfo
[0..1]	[0..*] DiagramObjects	DiagramObject	inherited from: IdentifiedObject
[1..1]	[0..*] Names	Name	inherited from: IdentifiedObject

6.5.20 WireInsulationKind enumeration

Kind of wire insulation.

Table 98 shows all literals of WireInsulationKind.

Table 98 – Literals of AssetInfo::WireInsulationKind

literal	description
asbestosAndVarnishedCambric	Asbestos and varnished cambric wire insulation.
butyl	Butyl wire insulation.
ethylenePropyleneRubber	Ethylene propylene rubber wire insulation.
highMolecularWeightPolyethylene	High nolecular weight polyethylene wire insulation.
treeResistantHighMolecularWeightPolyethylene	Tree resistant high molecular weight polyethylene wire insulation.
lowCapacitanceRubber	Low capacitance rubber wire insulation.
oilPaper	Oil paper wire insulation.
ozoneResistantRubber	Ozone resistant rubber wire insulation.
beltedPilc	Belted pilc wire insulation.
unbeltedPilc	Unbelted pilc wire insulation.
rubber	Rubber wire insulation.
siliconRubber	Silicon rubber wire insulation.
varnishedCambricCloth	Varnished cambric cloth wire insulation.
varnishedDacronGlass	Varnished dacron glass wire insulation.
crosslinkedPolyethylene	Crosslinked polyethylene wire insulation.
treeRetardantCrosslinkedPolyethylene	Tree retardant crosslinked polyethylene wire insulation.
highPressureFluidFilled	High pressure fluid filled wire insulation.
other	Other kind of wire insulation.

6.5.21 WireMaterialKind enumeration

Kind of wire material.

Table 99 shows all literals of WireMaterialKind.

Table 99 – Literals of AssetInfo::WireMaterialKind

literal	description
copper	Copper wire.
steel	Steel wire.
aluminum	Aluminum wire.
aluminumSteel	Aluminum-steel wire.
acsr	Aluminum conductor steel reinforced.
aluminumAlloy	Aluminum-alloy wire.
aluminumAlloySteel	Aluminum-alloy-steel wire.
aaac	Aluminum-alloy conductor steel reinforced.
other	Other wire material.

6.5.22 WirePosition

Identification, spacing and configuration of the wires of a conductor with respect to a structure.

Table 100 shows all attributes of WirePosition.

Table 100 – Attributes of AssetInfo::WirePosition

name	type	description
phase	SinglePhaseKind	Single phase or neutral designation for the wire with this position.
xCoord	Displacement	Signed horizontal distance from the wire at this position to a common reference point.
yCoord	Displacement	Signed vertical distance from the wire at this position: above ground (positive value) or burial depth below ground (negative value).
aliasName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject

Table 101 shows all association ends of WirePosition with other classes.

Table 101 – Association ends of AssetInfo::WirePosition with other classes

[mult from]	[mult to] name	type	description
[1..*]	[0..1] WireSpacingInfo	WireSpacingInfo	Wire spacing data this wire position belongs to.
[0..1]	[0..*] DiagramObjects	DiagramObject	inherited from: IdentifiedObject
[1..1]	[0..*] Names	Name	inherited from: IdentifiedObject

6.5.23 WireSpacingInfo

Wire spacing data that associates multiple wire positions with the line segment, and allows to calculate line segment impedances. Number of phases can be derived from the number of associated wire positions whose phase is not neutral.

Table 102 shows all attributes of WireSpacingInfo.

Table 102 – Attributes of AssetInfo::WireSpacingInfo

name	type	description
isCable	Boolean	If true, this spacing data describes a cable.
phaseWireCount	Integer	Number of wire sub-conductors in the symmetrical bundle (typically between 1 and 4).
phaseWireSpacing	Length	Distance between wire sub-conductors in a symmetrical bundle.
usage	WireUsageKind	Usage of the associated wires.
aliasName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject

Table 103 shows all association ends of WireSpacingInfo with other classes.

Table 103 – Association ends of AssetInfo::WireSpacingInfo with other classes

[mult from]	[mult to] name	type	description
[0..1]	[0..*] Impedances	PerLengthImpedance	All impedances calculated from this wire spacing datasheet.
[0..1]	[1..*] WirePositions	WirePosition	All positions of single wires (phase or neutral) making the conductor.
[0..1]	[0..*] PowerSystemResource	PowerSystemResource	inherited from: AssetInfo
[0..1]	[0..*] Assets	Asset	inherited from: AssetInfo
[0..1]	[0..1] AssetModel	AssetModel	inherited from: AssetInfo
[0..1]	[0..*] DiagramObjects	DiagramObject	inherited from: IdentifiedObject
[1..1]	[0..*] Names	Name	inherited from: IdentifiedObject

6.5.24 WireUsageKind enumeration

Kind of wire usage.

Table 104 shows all literals of WireUsageKind.

Table 104 – Literals of AssetInfo::WireUsageKind

literal	description
transmission	Wire is used in extra-high voltage or high voltage network.
distribution	Wire is used in medium voltage network.
secondary	Wire is used in low voltage circuit.
other	Other kind of wire usage.

6.6 Package Work

6.6.1 General

This package contains the core information classes that support work management and network extension planning applications.

Figure 38 shows class diagram WorkInheritance.

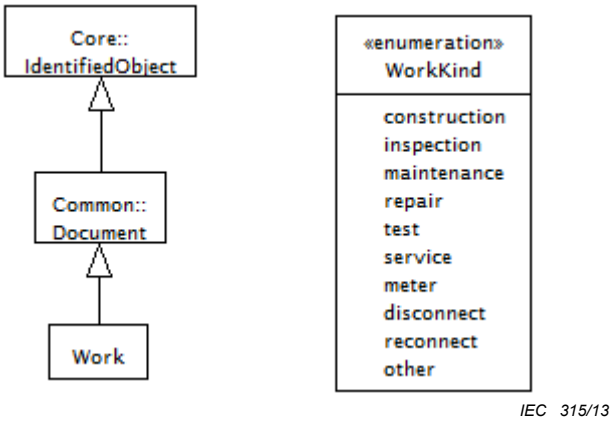


Figure 38 – Class diagram Work::WorkInheritance

This diagram shows the inheritance hierarchy for normative classes from this package, as well as enumerations and compound types.

Figure 39 shows class diagram WorkOverview.

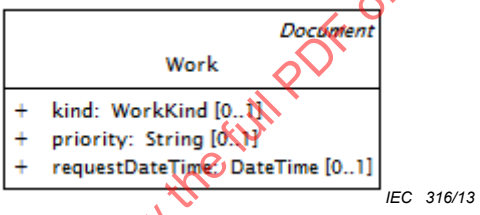


Figure 39 – Class diagram Work::WorkOverview

This diagram shows normative classes from this package.

6.6.2 WorkKind enumeration

Kind of work.

Table 105 shows all literals of WorkKind.

Table 105 – Literals of Work::WorkKind

literal	description
construction	Construction work.
inspection	Inspection work.
maintenance	Maintenance work.
repair	Repair work.
test	Test work.
service	Service work.
meter	Meter work.
disconnect	Disconnect work.
reconnect	Reconnect work.
other	Other kind of work.

6.6.3 Work

Document used to request, initiate, track and record work.

Table 106 shows all attributes of Work.

Table 106 – Attributes of Work::Work

name	type	description
kind	WorkKind	Kind of work.
priority	String	Priority of work.
requestDateTime	DateTime	Date and time work was requested.
type	String	inherited from: Document
createdDateTime	DateTime	inherited from: Document
lastModifiedDateTime	DateTime	inherited from: Document
revisionNumber	String	inherited from: Document
electronicAddress	ElectronicAddress	inherited from: Document
subject	String	inherited from: Document
title	String	inherited from: Document
docStatus	Status	inherited from: Document
status	Status	inherited from: Document
aliasName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject

Table 107 shows all association ends of Work with other classes.

Table 107 – Association ends of Work::Work with other classes

[mult from]	[mult to] name	type	description
[0..*]	[0..*] Customers	Customer	All the customers for which this work is performed.
[0..1]	[0..*] ConfigurationEvents	ConfigurationEvent	inherited from: Document
[0..1]	[0..*] DiagramObjects	DiagramObject	inherited from: IdentifiedObject
[1..1]	[0..*] Names	Name	inherited from: IdentifiedObject

6.7 Package Customers

6.7.1 General

This package contains the core information classes that support customer billing applications.

Figure 40 shows class diagram CustomersInheritance.

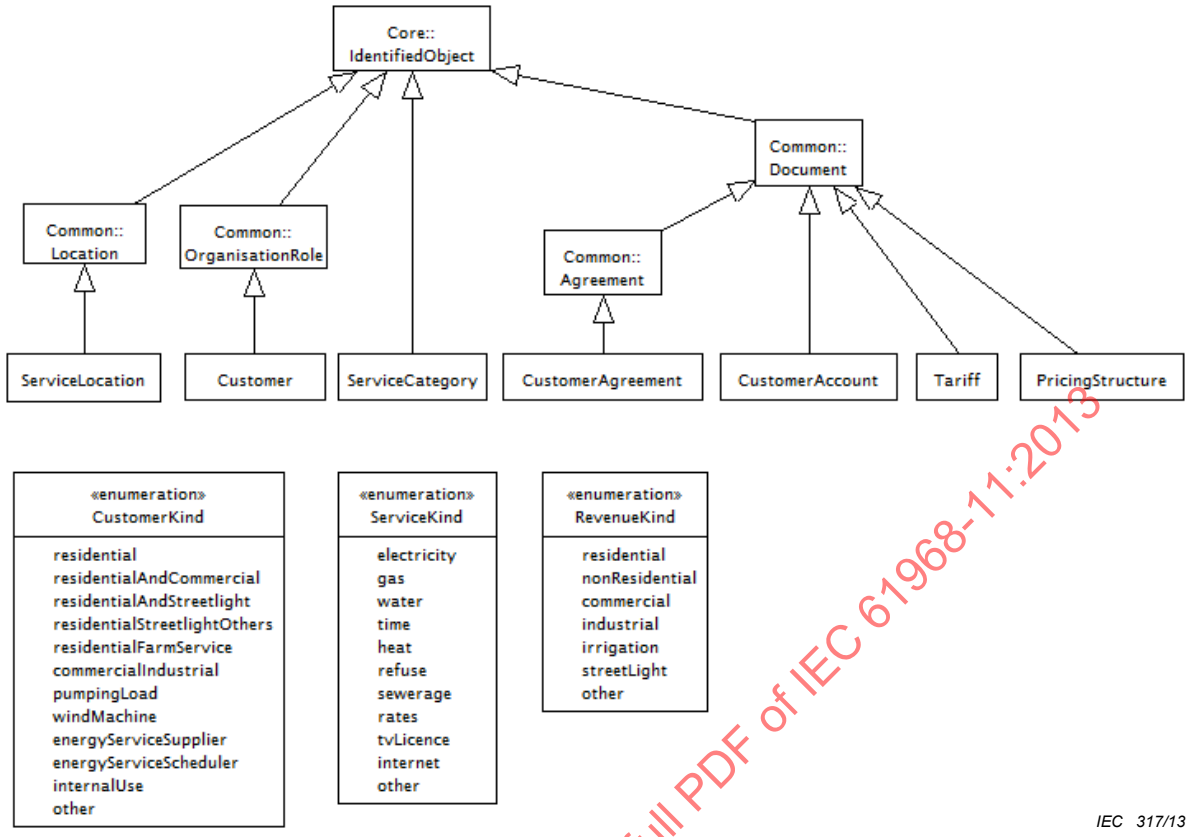


Figure 40 – Class diagram Customers::CustomersInheritance

This diagram shows the inheritance hierarchy for normative classes from this package, as well as enumerations and compound types.

Figure 41 shows class diagram CustomersOverview.

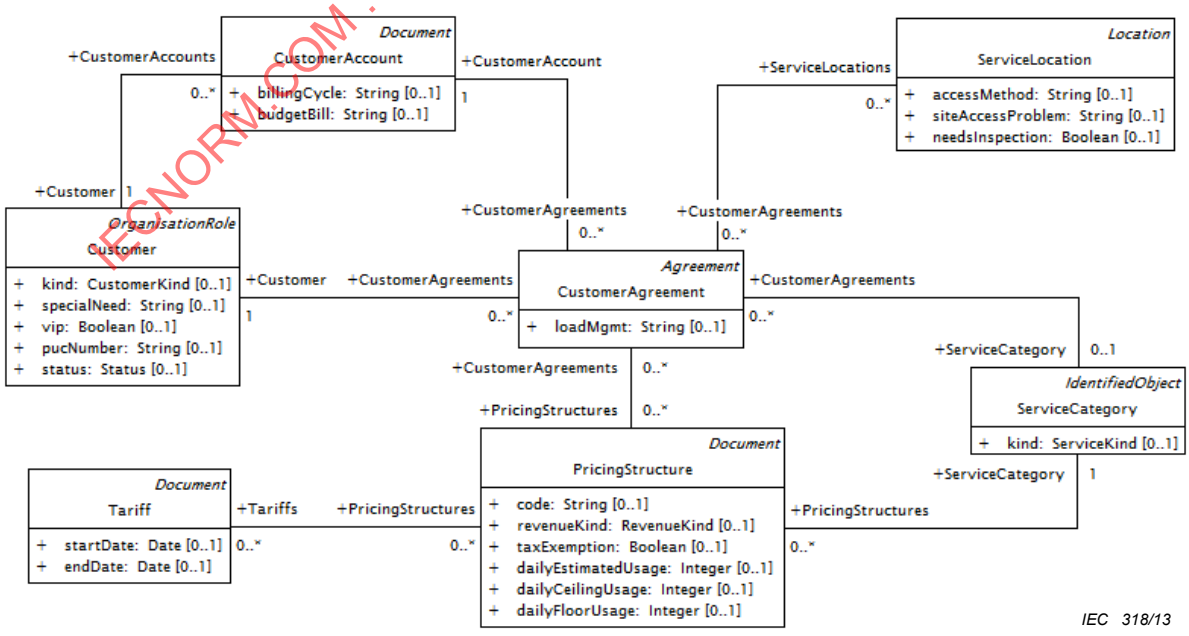


Figure 41 – Class diagram Customers::CustomersOverview

This diagram shows normative classes from this package.

6.7.2 CustomerKind enumeration

Kind of customer.

Table 108 shows all literals of CustomerKind.

Table 108 – Literals of Customers::CustomerKind

literal	description
residential	Residential customer.
residentialAndCommercial	Residential and commercial customer.
residentialAndStreetlight	Residential and streetlight customer.
residentialStreetlightOthers	Residential streetlight or other related customer.
residentialFarmService	Residential farm service customer.
commercialIndustrial	Commercial industrial customer.
pumpingLoad	Pumping load customer.
windMachine	Wind machine customer.
energyServiceSupplier	Customer as energy service supplier.
energyServiceScheduler	Customer as energy service scheduler.
internalUse	Internal use customer.
other	Other kind of customer.

6.7.3 RevenueKind enumeration

Accounting classification of the type of revenue collected for the customer agreement, typically used to break down accounts for revenue accounting.

Table 109 shows all literals of RevenueKind.

Table 109 – Literals of Customers::RevenueKind

literal	description
residential	Residential revenue.
nonResidential	Non-residential revenue.
commercial	Commercial revenue.
industrial	Industrial revenue.
irrigation	Irrigation revenue.
streetLight	Streetlight revenue.
other	Other revenue kind.

6.7.4 ServiceKind enumeration

Kind of service.

Table 110 shows all literals of ServiceKind.

Table 110 – Literals of Customers::ServiceKind

literal	description
electricity	Electricity service.
gas	Gas service.
water	Water service.
time	Time service.
heat	Heat service.
refuse	Refuse (waster) service.
sewerage	Sewerage service.
rates	Rates (e.g. tax, charge, toll, duty, tariff, etc.) service.
tvLicence	TV license service.
internet	Internet service.
other	Other kind of service.

6.7.5 Customer

Organisation receiving services from service supplier.

Table 111 shows all attributes of Customer.

Table 111 – Attributes of Customers::Customer

name	type	description
kind	CustomerKind	Kind of customer.
specialNeed	String	True if customer organisation has special service needs such as life support, hospitals, etc.
vip	Boolean	True if this is an important customer. Importance is for matters different than those in 'specialNeed' attribute.
pucNumber	String	(if applicable) Public utilities commission (PUC) identification number.
status	Status	Status of this customer.
aliasName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject

Table 112 shows all association ends of Customer with other classes.

Table 112 – Association ends of Customers::Customer with other classes

[mult from]	[mult to] name	type	description
[0..*]	[0..*] Works	Work	All the works performed for this customer.
[0..1]	[0..*] EndDevices	EndDevice	All end devices of this customer.
[1..1]	[0..*] CustomerAccounts	CustomerAccount	All accounts of this customer.
[1..1]	[0..*] CustomerAgreements	CustomerAgreement	All agreements of this customer.
[0..1]	[0..*] ConfigurationEvents	ConfigurationEvent	inherited from: OrganisationRole
[0..*]	[0..1] Organisation	Organisation	inherited from: OrganisationRole

[mult from]	[mult to] name	type	description
[0..1]	[0..*] DiagramObjects	DiagramObject	inherited from: IdentifiedObject
[1..1]	[0..*] Names	Name	inherited from: IdentifiedObject

6.7.6 CustomerAccount

Assignment of a group of products and services purchased by the customer through a customer agreement, used as a mechanism for customer billing and payment. It contains common information from the various types of customer agreements to create billings (invoices) for a customer and receive payment.

Table 113 shows all attributes of CustomerAccount.

Table 113 – Attributes of Customers::CustomerAccount

name	type	description
billingCycle	String	Cycle day on which the associated customer account will normally be billed, used to determine when to produce the billing.
budgetBill	String	Budget bill code.
type	String	inherited from: Document
createdDateTime	DateTime	inherited from: Document
lastModifiedDateTime	DateTime	inherited from: Document
revisionNumber	String	inherited from: Document
electronicAddress	ElectronicAddress	inherited from: Document
subject	String	inherited from: Document
title	String	inherited from: Document
docStatus	Status	inherited from: Document
status	Status	inherited from: Document
aliasName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject

Table 114 shows all association ends of CustomerAccount with other classes.

Table 114 – Association ends of Customers::CustomerAccount with other classes

[mult from]	[mult to] name	type	description
[0..1]	[0..*] PaymentTransactions	Transaction	All payment transactions for this customer account.
[1..1]	[0..*] CustomerAgreements	CustomerAgreement	All agreements for this customer account.
[0..*]	[1..1] Customer	Customer	Customer owning this account.
[0..1]	[0..*] ConfigurationEvents	ConfigurationEvent	inherited from: Document
[0..1]	[0..*] DiagramObjects	DiagramObject	inherited from: IdentifiedObject
[1..1]	[0..*] Names	Name	inherited from: IdentifiedObject

6.7.7 CustomerAgreement

Agreement between the customer and the service supplier to pay for service at a specific service location. It records certain billing information about the type of service provided at the service location and is used during charge creation to determine the type of service.

Table 115 shows all attributes of CustomerAgreement.

Table 115 – Attributes of Customers::CustomerAgreement

name	type	description
loadMgmt	String	Load management code.
signDate	Date	inherited from: Agreement
validityInterval	DateTimeInterval	inherited from: Agreement
type	String	inherited from: Document
createdDateTime	DateTime	inherited from: Document
lastModifiedDateTime	DateTime	inherited from: Document
revisionNumber	String	inherited from: Document
electronicAddress	ElectronicAddress	inherited from: Document
subject	String	inherited from: Document
title	String	inherited from: Document
docStatus	Status	inherited from: Document
status	Status	inherited from: Document
aliasName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject

Table 116 shows all association ends of CustomerAgreement with other classes.

Table 116 – Association ends of Customers::CustomerAgreement with other classes

[mult from]	[mult to] name	type	description
[0..*]	[0..*] ServiceLocations	ServiceLocation	All service locations regulated by this customer agreement.
[0..1]	[0..*] AuxiliaryAgreements	AuxiliaryAgreement	All (non-service related) auxiliary agreements that refer to this customer agreement.
[0..*]	[0..*] PricingStructures	PricingStructure	All pricing structures applicable to this customer agreement.
[0..1]	[0..*] UsagePoints	UsagePoint	All service delivery points regulated by this customer agreement.
[0..*]	[0..1] ServiceCategory	ServiceCategory	Service category for this agreement.
[0..*]	[1..1] Customer	Customer	Customer for this agreement.
[0..*]	[1..1] CustomerAccount	CustomerAccount	Customer account owning this agreement.
[0..*]	[0..*] DemandResponsePrograms	DemandResponseProgram	All demand response programs the customer is enrolled in through this customer agreement.
[0..1]	[0..*] MeterReadings	MeterReading	(could be deprecated in the future) All meter readings for this customer agreement.

[mult from]	[mult to] name	type	description
[0..*]	[1..1] ServiceSupplier	ServiceSupplier	Service supplier for this customer agreement.
[0..1]	[0..*] ConfigurationEvents	ConfigurationEvent	inherited from: Document
[0..1]	[0..*] DiagramObjects	DiagramObject	inherited from: IdentifiedObject
[1..1]	[0..*] Names	Name	inherited from: IdentifiedObject

6.7.8 PricingStructure

Grouping of pricing components and prices used in the creation of customer charges and the eligibility criteria under which these terms may be offered to a customer. The reasons for grouping include state, customer classification, site characteristics, classification (i.e. fee price structure, deposit price structure, electric service price structure, etc.) and accounting requirements.

Table 117 shows all attributes of PricingStructure.

Table 117 – Attributes of Customers::PricingStructure

name	type	description
code	String	Unique user-allocated key for this pricing structure, used by company representatives to identify the correct price structure for allocating to a customer. For rate schedules it is often prefixed by a state code.
revenueKind	RevenueKind	(accounting) Kind of revenue, often used to determine the grace period allowed, before collection actions are taken on a customer (grace periods vary between revenue classes).
taxExemption	Boolean	True if this pricing structure is not taxable.
dailyEstimatedUsage	Integer	Used in place of actual computed estimated average when history of usage is not available, and typically manually entered by customer accounting.
dailyCeilingUsage	Integer	Absolute maximum valid non-demand usage quantity used in validating a customer's billed non-demand usage.
dailyFloorUsage	Integer	Absolute minimum valid non-demand usage quantity used in validating a customer's billed non-demand usage.
type	String	inherited from: Document
createdDateTime	DateTime	inherited from: Document
lastModifiedDateTime	DateTime	inherited from: Document
revisionNumber	String	inherited from: Document
electronicAddress	ElectronicAddress	inherited from: Document
subject	String	inherited from: Document
title	String	inherited from: Document
docStatus	Status	inherited from: Document
status	Status	inherited from: Document
aliasName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject

Table 118 shows all association ends of PricingStructure with other classes.

Table 118 – Association ends of Customers::PricingStructure with other classes

[mult from]	[mult to] name	type	description
[0..*]	[0..*] UsagePoints	UsagePoint	All service delivery points (with prepayment meter running as a stand-alone device, with no CustomerAgreement or Customer) to which this pricing structure applies.
[0..1]	[0..*] Transactions	Transaction	All transactions applying this pricing structure.
[0..*]	[0..*] Tariffs	Tariff	All tariffs used by this pricing structure.
[0..*]	[0..*] CustomerAgreements	CustomerAgreement	All customer agreements with this pricing structure.
[0..*]	[1..1] ServiceCategory	ServiceCategory	Service category to which this pricing structure applies.
[0..1]	[0..*] ConfigurationEvents	ConfigurationEvent	inherited from: Document
[0..1]	[0..*] DiagramObjects	DiagramObject	inherited from: IdentifiedObject
[1..1]	[0..*] Names	Name	inherited from: IdentifiedObject

6.7.9 ServiceCategory

Category of service provided to the customer.

Table 119 shows all attributes of ServiceCategory.

Table 119 – Attributes of Customers::ServiceCategory

name	type	description
kind	ServiceKind	Kind of service.
aliasName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject

Table 120 shows all association ends of ServiceCategory with other classes.

Table 120 – Association ends of Customers::ServiceCategory with other classes

[mult from]	[mult to] name	type	description
[0..1]	[0..*] ConfigurationEvents	ConfigurationEvent	All configuration events created for this service category.
[1..1]	[0..*] PricingStructures	PricingStructure	All pricing structures applicable to this service category.
[0..1]	[0..*] CustomerAgreements	CustomerAgreement	All customer agreements with this service category.
[0..1]	[0..*] UsagePoints	UsagePoint	All usage points that deliver this category of service.
[0..1]	[0..*] DiagramObjects	DiagramObject	inherited from: IdentifiedObject
[1..1]	[0..*] Names	Name	inherited from: IdentifiedObject

6.7.10 ServiceLocation

A real estate location, commonly referred to as premises.

Table 121 shows all attributes of ServiceLocation.

Table 121 – Attributes of Customers::ServiceLocation

name	type	description
accessMethod	String	Method for the service person to access this service location. For example, a description of where to obtain a key if the facility is unmanned and secured.
siteAccessProblem	String	Problems previously encountered when visiting or performing work on this location. Examples include: bad dog, violent customer, verbally abusive occupant, obstructions, safety hazards, etc.
needsInspection	Boolean	True if inspection is needed of facilities at this service location. This could be requested by a customer, due to suspected tampering, environmental concerns (e.g., a fire in the vicinity), or to correct incompatible data.
type	String	inherited from: Location
mainAddress	StreetAddress	inherited from: Location
secondaryAddress	StreetAddress	inherited from: Location
phone1	TelephoneNumber	inherited from: Location
phone2	TelephoneNumber	inherited from: Location
electronicAddress	ElectronicAddress	inherited from: Location
geoInfoReference	String	inherited from: Location
direction	String	inherited from: Location
status	Status	inherited from: Location
aliasName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject

Table 122 shows all association ends of ServiceLocation with other classes.

Table 122 – Association ends of Customers::ServiceLocation with other classes

[mult from]	[mult to] name	type	description
[0..1]	[0..*] EndDevices	EndDevice	All end devices that measure the service delivered to this service location.
[0..1]	[0..*] UsagePoints	UsagePoint	All usage points delivering service (of the same type) to this service location.
[0..*]	[0..*] CustomerAgreements	CustomerAgreement	All customer agreements regulating this service location.
[0..1]	[0..*] PowerSystemResources	PowerSystemResource	inherited from: Location
[0..1]	[0..*] ConfigurationEvents	ConfigurationEvent	inherited from: Location
[0..*]	[0..1] CoordinateSystem	CoordinateSystem	inherited from: Location
[1..1]	[0..*] PositionPoints	PositionPoint	inherited from: Location
[0..1]	[0..*] Assets	Asset	inherited from: Location
[0..1]	[0..*] DiagramObjects	DiagramObject	inherited from: IdentifiedObject
[1..1]	[0..*] Names	Name	inherited from: IdentifiedObject

6.7.11 Tariff

Document, approved by the responsible regulatory agency, listing the terms and conditions, including a schedule of prices, under which utility services will be provided. It has a unique number within the state or province. For rate schedules it is frequently allocated by the affiliated Public utilities commission (PUC).

Table 123 shows all attributes of Tariff.

Table 123 – Attributes of Customers::Tariff

name	type	description
startDate	Date	Date tariff was activated.
endDate	Date	(if tariff became inactive) Date tariff was terminated.
type	String	inherited from: Document
createdDateTime	DateTime	inherited from: Document
lastModifiedDateTime	DateTime	inherited from: Document
revisionNumber	String	inherited from: Document
electronicAddress	ElectronicAddress	inherited from: Document
subject	String	inherited from: Document
title	String	inherited from: Document
docStatus	Status	inherited from: Document
status	Status	inherited from: Document
aliasName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject

Table 124 shows all association ends of Tariff with other classes.

Table 124 – Association ends of Customers::Tariff with other classes

[mult from]	[mult to] name	type	description
[0..*]	[0..*] TariffProfiles	TariffProfile	All tariff profiles using this tariff.
[0..*]	[0..*] PricingStructures	PricingStructure	All pricing structures using this tariff.
[0..1]	[0..*] ConfigurationEvents	ConfigurationEvent	inherited from: Document
[0..1]	[0..*] DiagramObjects	DiagramObject	inherited from: IdentifiedObject
[1..1]	[0..*] Names	Name	inherited from: IdentifiedObject

6.8 Package Metering

6.8.1 General

This package contains the core information classes that support end device applications with specialized classes for metering and premise area network devices, and remote reading functions. These classes are generally associated with the point where a service is delivered to the customer.

Figure 42 shows class diagram MeteringInheritance.

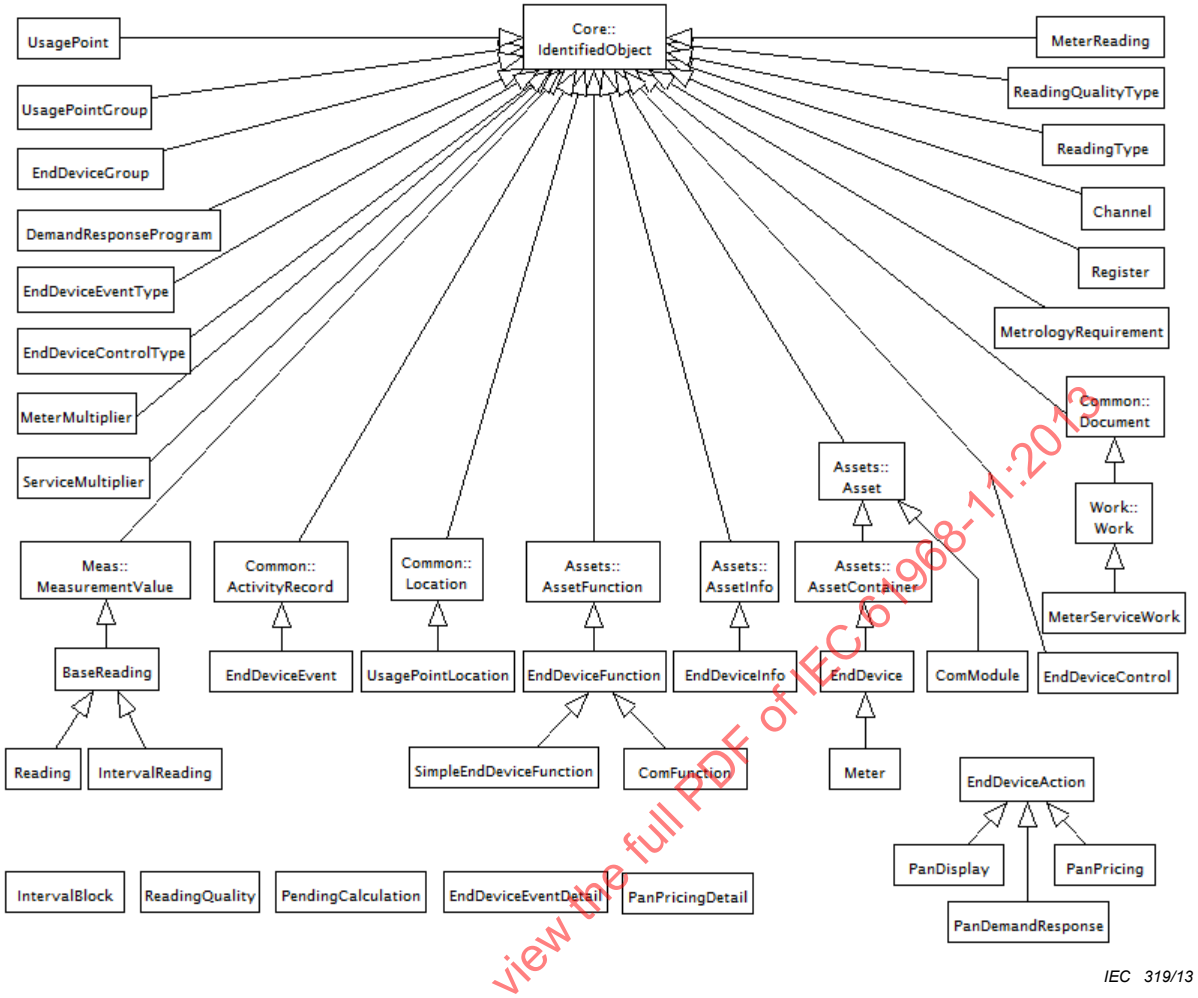


Figure 42 – Class diagram Metering::MeteringInheritance

This diagram shows the inheritance hierarchy for normative classes from this package.

Figure 43 shows class diagram MeteringDatatypes.

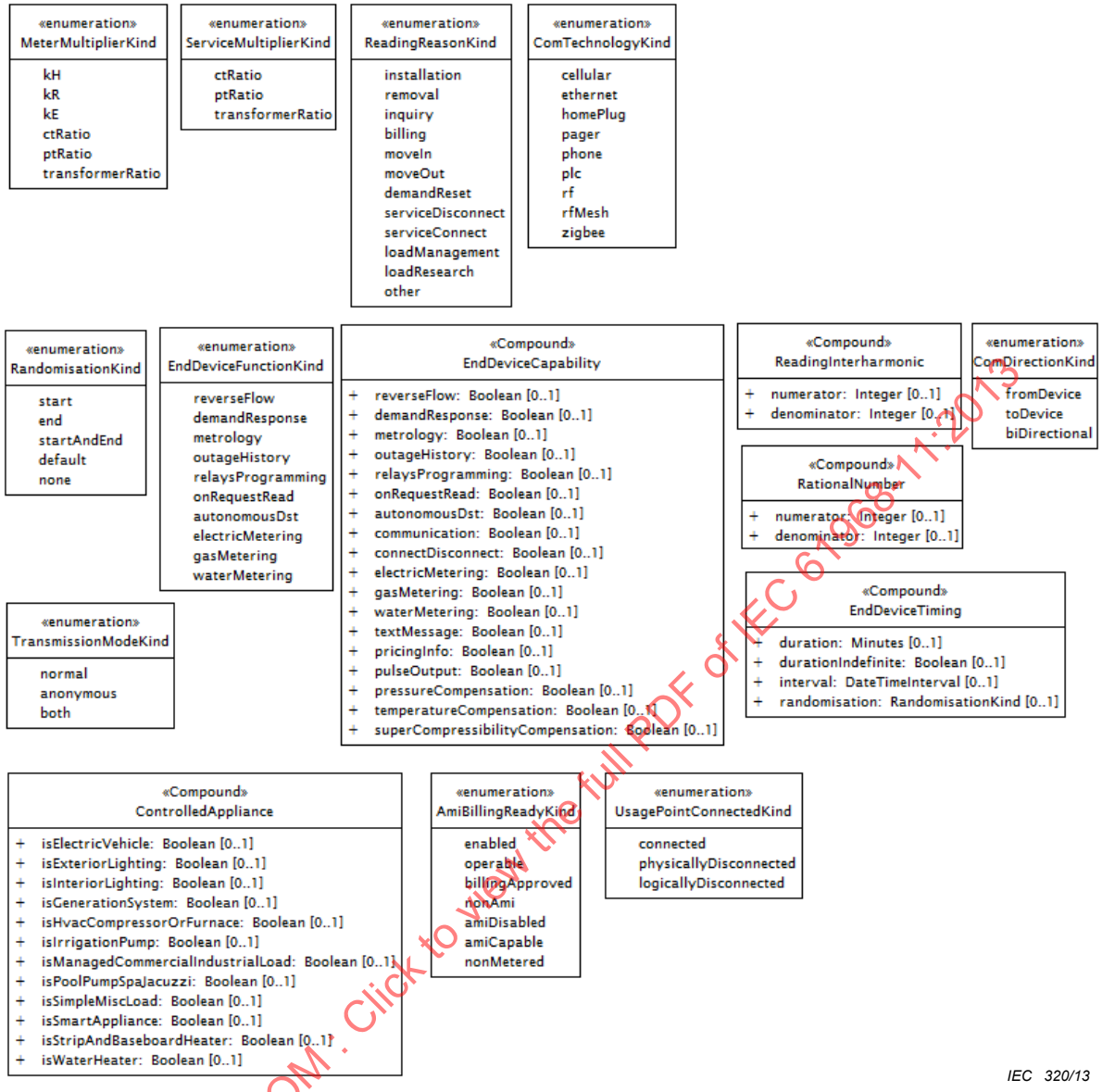


Figure 43 – Class diagram Metering::MeteringDatatypes

This diagram shows enumerations and compound types from this package.

Figure 44 shows class diagram MeteringOverviewShort.

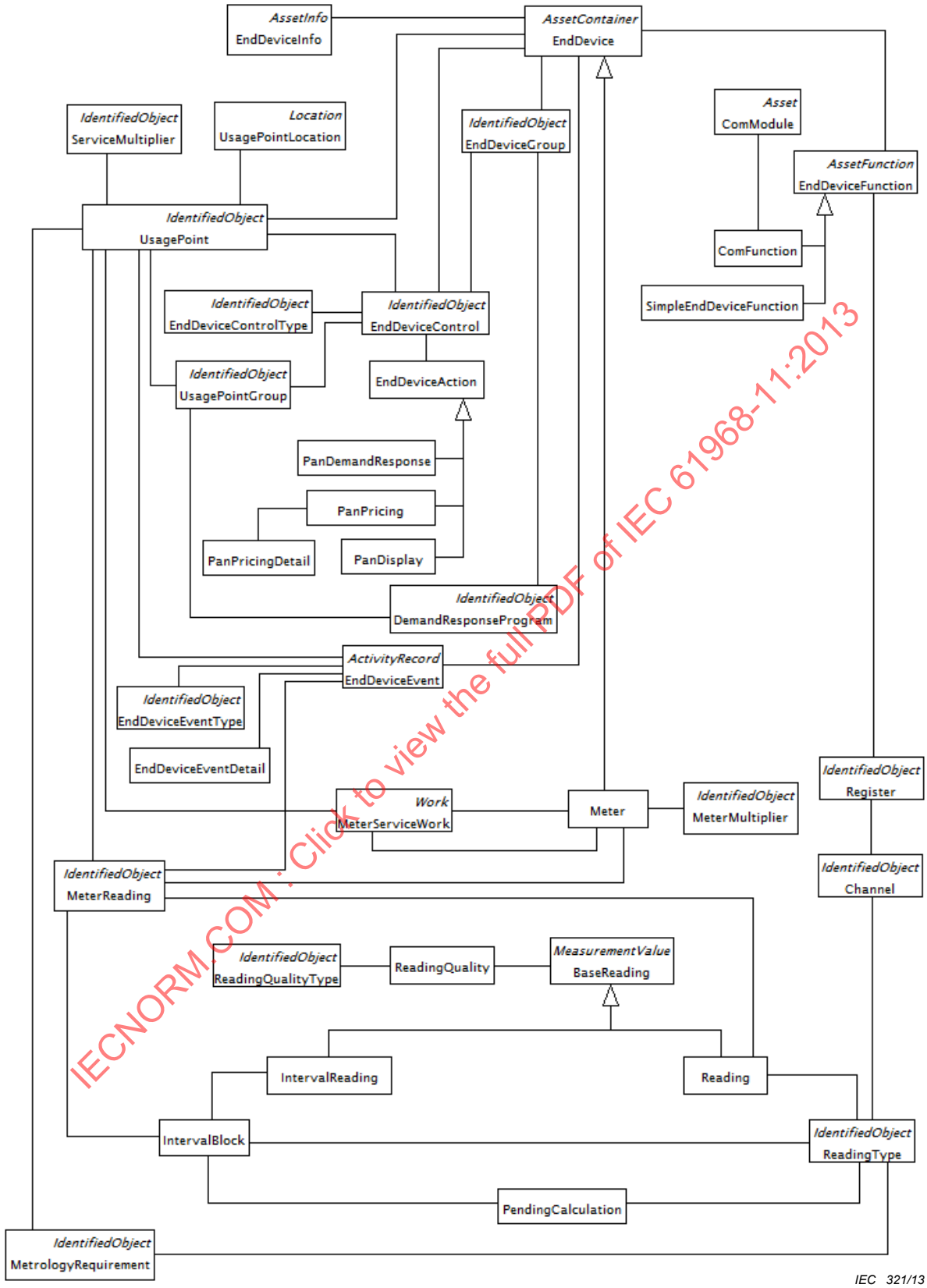


Figure 44 – Class diagram Metering::MeteringOverviewShort

This diagram shows relationships among normative classes from this package (without details on attributes and associations).

Figure 45 shows class diagram MeteringUsagePoints.

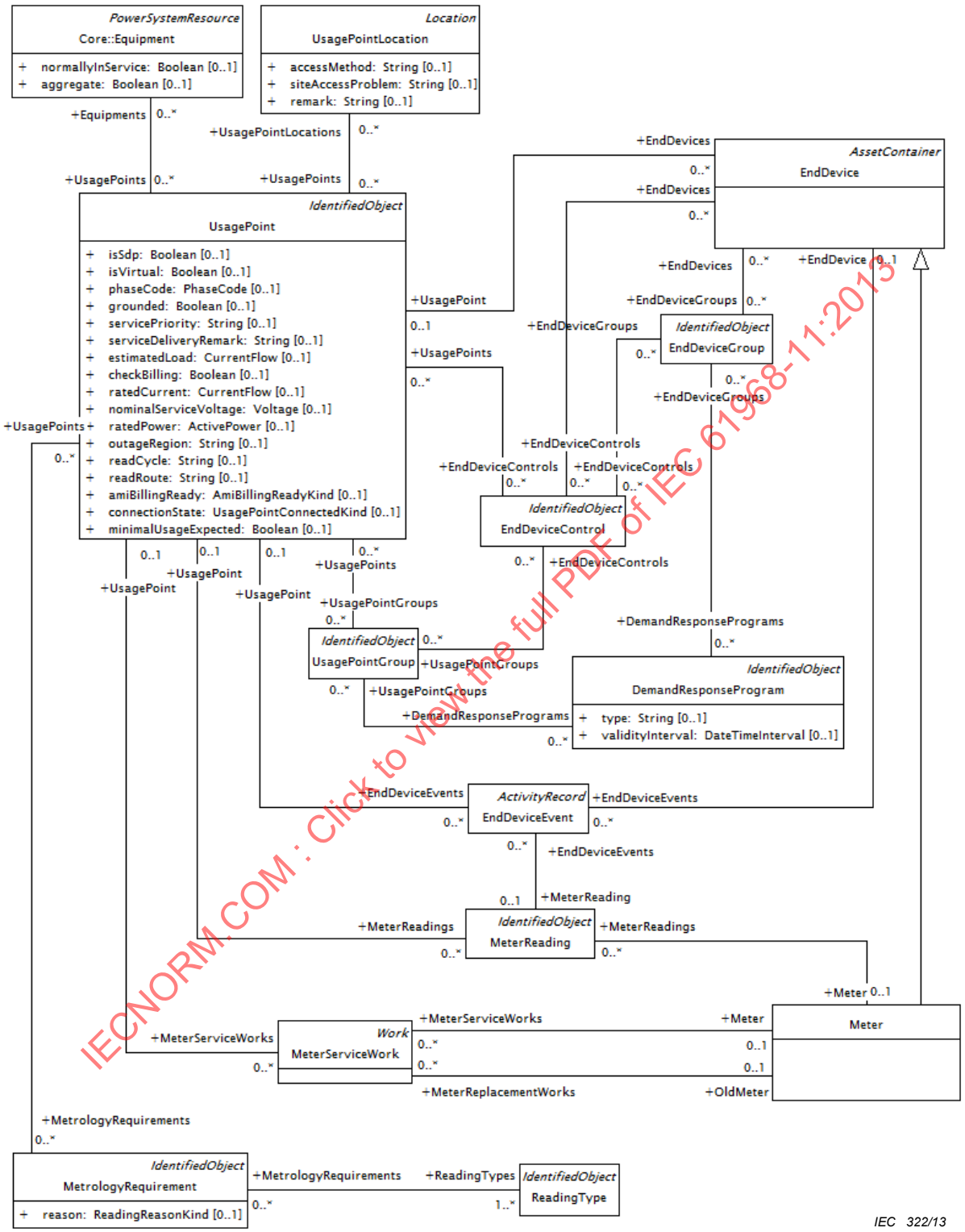


Figure 45 – Class diagram Metering::MeteringUsagePoints

This diagram shows the subset of normative classes from this package focused on usage points.

Figure 46 shows class diagram MeteringEndDevices.

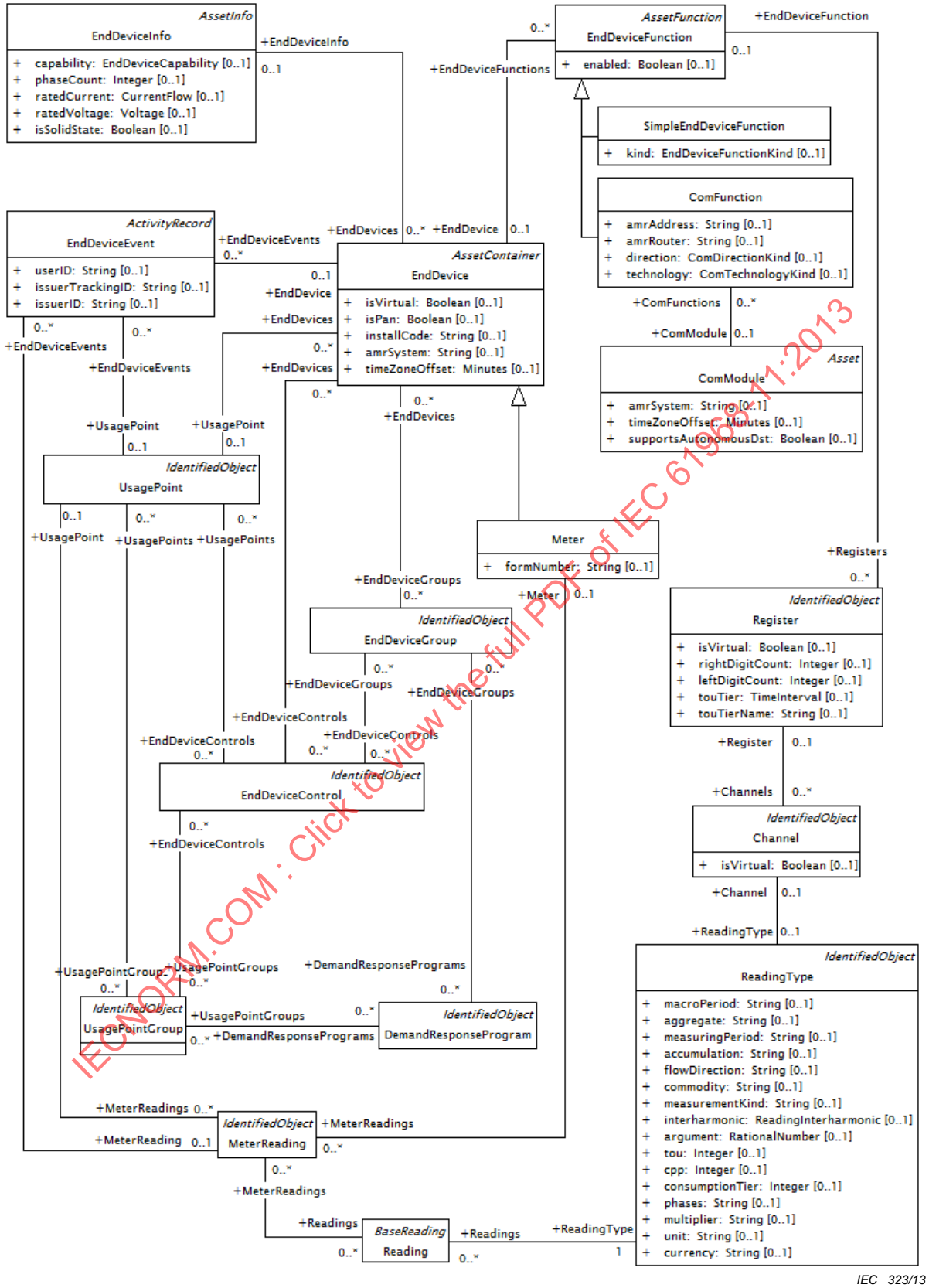


Figure 46 – Class diagram Metering::MeteringEndDevices

This diagram shows the subset of normative classes from this package focused on end devices.

Figure 47 shows class diagram MeteringConfigurationEvents.

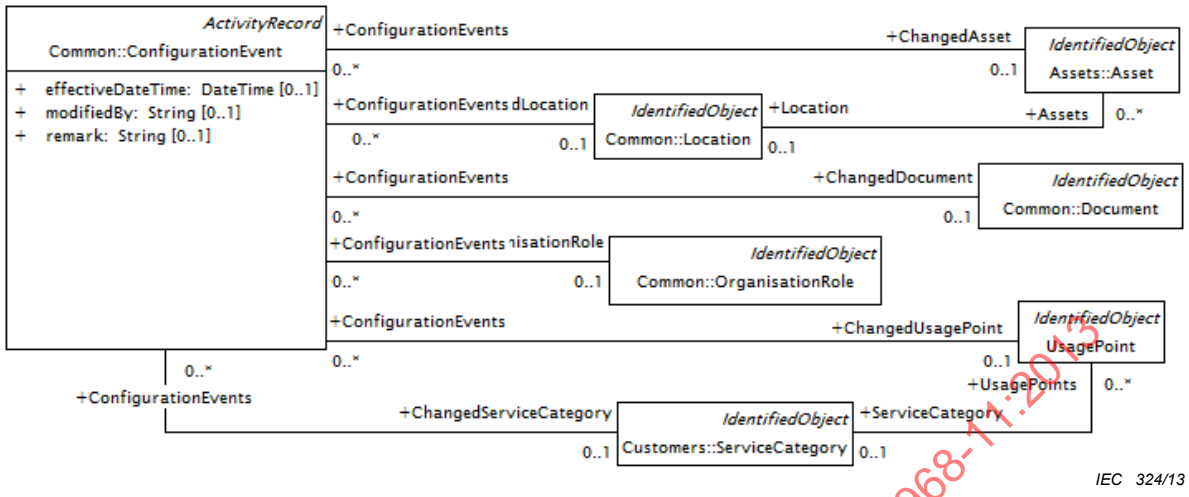


Figure 47 – Class diagram Metering::MeteringConfigurationEvents

This diagram shows the subset of normative classes from this package focused on configuration events.

Figure 48 shows class diagram MeteringMeterReadings.

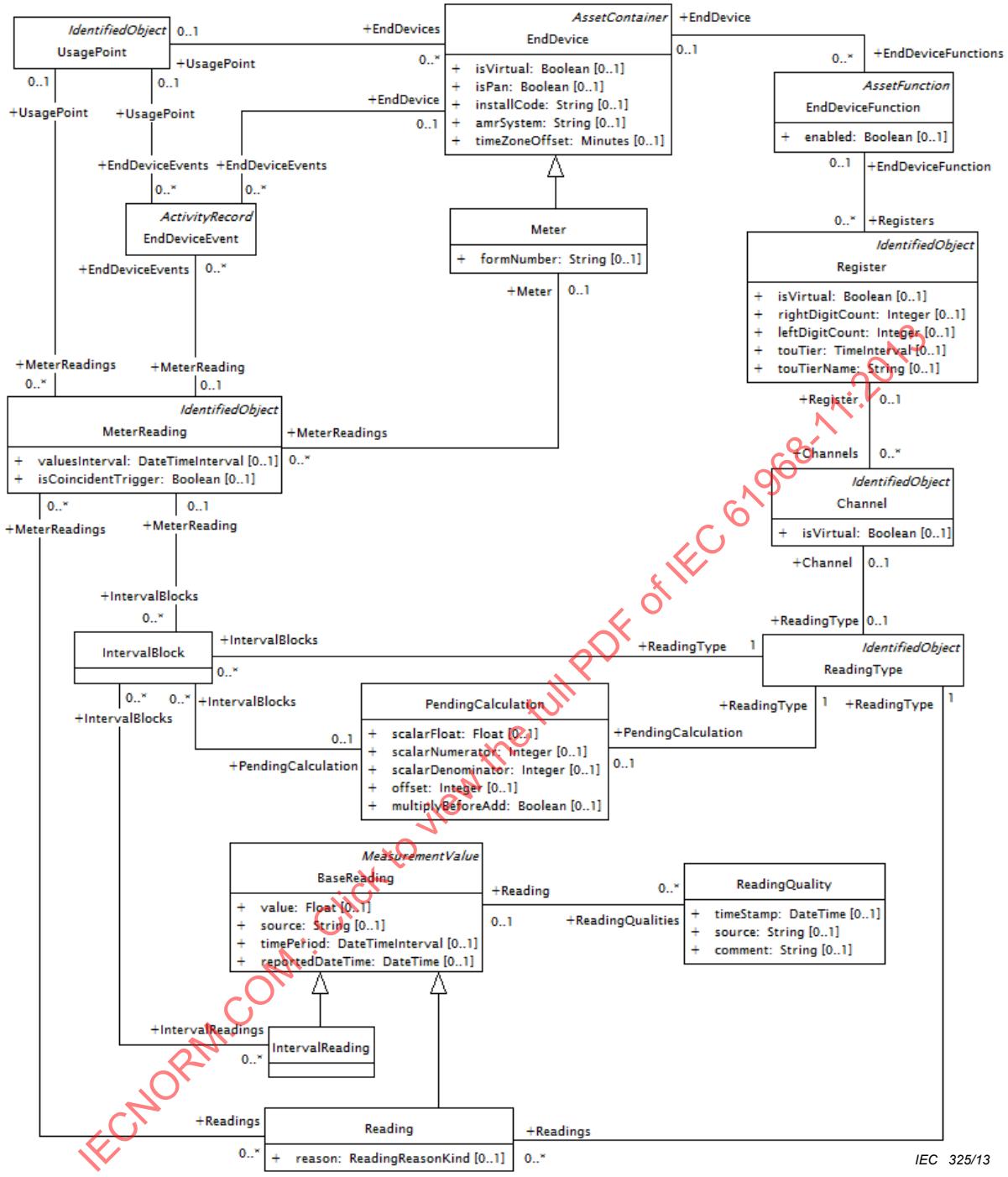


Figure 48 – Class diagram Metering::MeteringMeterReadings

This diagram shows the subset of normative classes from this package focused on meter readings.

Figure 49 shows class diagram MeteringEventsAndControls.

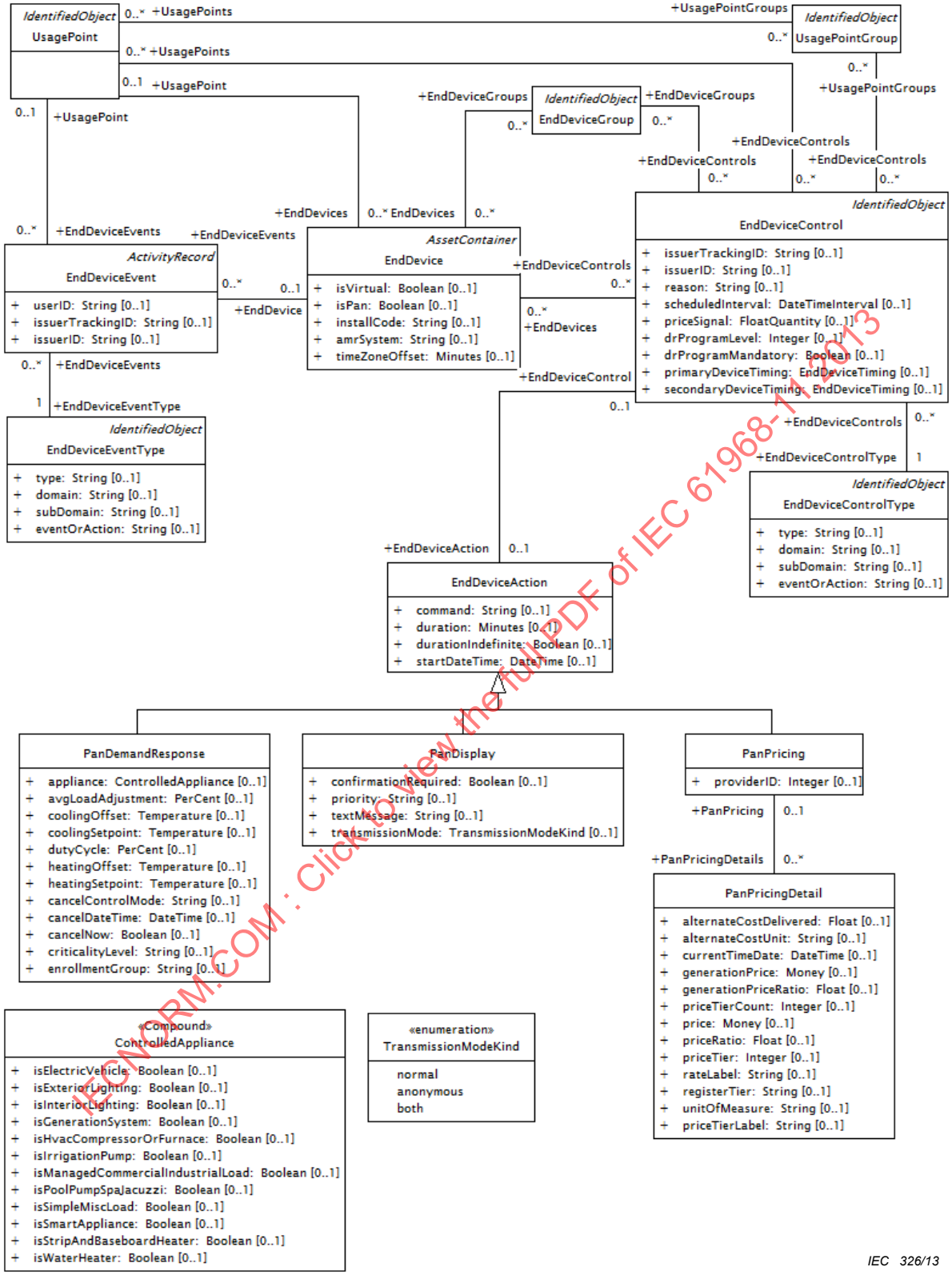


Figure 49 – Class diagram Metering::MeteringEventsAndControls

This diagram shows the subset of normative classes from this package focused on end device events, controls and actions.

Figure 50 shows class diagram MeteringMultipliers.

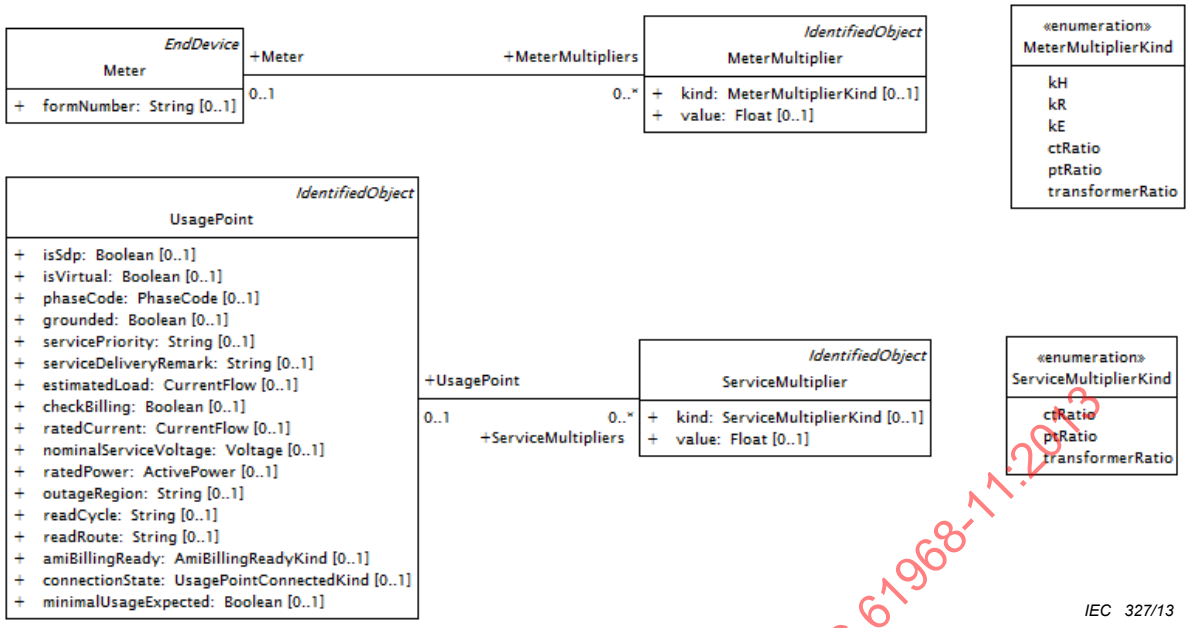


Figure 50 – Class diagram Metering::MeteringMultipliers

This diagrams shows the subset of normative classes from this package focused on multipliers.

Figure 51 shows class diagram MeteringTypes.

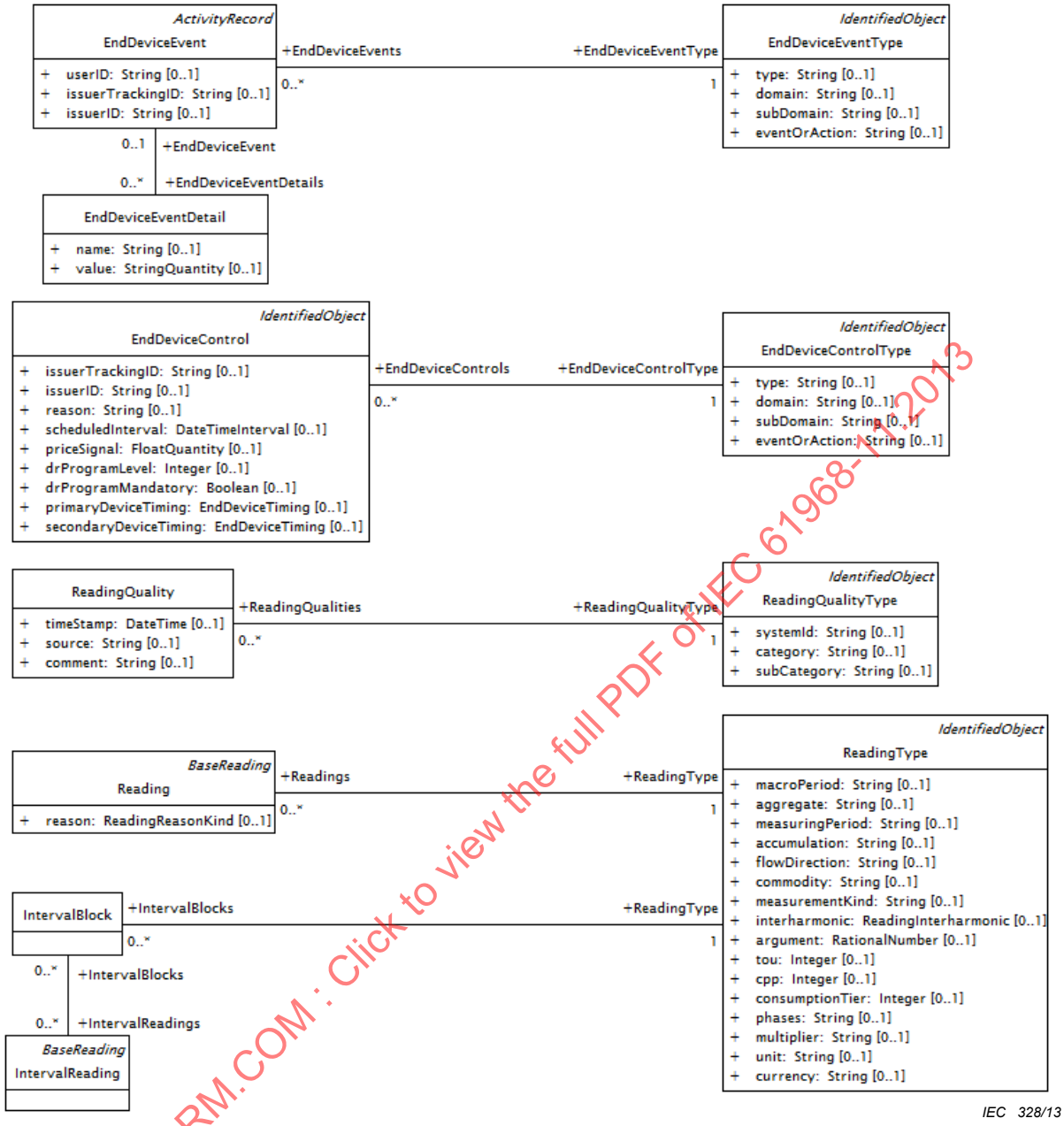


Figure 51 – Class diagram Metering::MeteringTypes

This diagram shows the subset of normative classes from this package focused on types used for end device events and controls, and for reading values quality and type.

6.8.2 AmiBillingReadyKind enumeration

Lifecycle states of the metering installation at a usage point with respect to readiness for billing via advanced metering infrastructure reads.

Table 125 shows all literals of AmiBillingReadyKind.

Table 125 – Literals of Metering::AmiBillingReadyKind

literal	description
enabled	Usage point is equipped with an AMI capable meter having communications capability.
operable	Usage point is equipped with an AMI capable meter that is functioning and communicating with the AMI network.
billingApproved	Usage point is equipped with an operating AMI capable meter and accuracy has been certified for billing purposes.
nonAmi	Usage point is equipped with a non AMI capable meter.
amiDisabled	Usage point is equipped with an AMI capable meter; however, the AMI functionality has been disabled or is not being used.
amiCapable	Usage point is equipped with an AMI capable meter that is not yet currently equipped with a communications module.
nonMetered	Usage point is not currently equipped with a meter.

6.8.3 ComDirectionKind enumeration

Kind of communication direction.

Table 126 shows all literals of ComDirectionKind.

Table 126 – Literals of Metering::ComDirectionKind

literal	description
fromDevice	Communication is from device.
toDevice	Communication is to device.
biDirectional	Communication with the device is bi-directional.

6.8.4 ComTechnologyKind enumeration

Kind of communication technology.

Table 127 shows all literals of ComTechnologyKind.

Table 127 – Literals of Metering::ComTechnologyKind

literal	description
cellular	Communicates using a public cellular radio network. A specific variant of 'rf'.
ethernet	Communicates using one or more of a family of frame-based computer networking technologies conforming to the IEEE 802.3 standard.
homePlug	Communicates using power line communication technologies conforming to the standards established by the HomePlug Powerline Alliance. A specific variant of 'plc'.
pager	Communicates using a public one-way or two-way radio-based paging network. A specific variant of 'rf'.
phone	Communicates using a basic, wireline telephone system.
plc	Communicates using power line communication technologies.
rf	Communicates using private or public radio-based technology.
rfMesh	Communicates using a mesh radio technology. A specific variant of 'rf'.
zigbee	Communicates using radio communication technologies conforming to the standards established by the ZigBee. A specific variant of 'rf'.

6.8.5 EndDeviceFunctionKind enumeration

Kind of end device function.

Table 128 shows all literals of EndDeviceFunctionKind.

Table 128 – Literals of Metering::EndDeviceFunctionKind

literal	description
reverseFlow	Detection and monitoring of reverse flow.
demandResponse	Demand response functions.
metrology	Presentation of metered values to a user or another system (always a function of a meter, but might not be supported by a load control unit).
outageHistory	Reporting historical power interruption data.
relaysProgramming	Support for one or more relays that may be programmable in the meter (and tied to TOU, time pulse, load control or other functions).
onRequestRead	On-request reads.
autonomousDst	Autonomous application of daylight saving time (DST).
electricMetering	Electricity metering.
gasMetering	Gas metering.
waterMetering	Water metering.

6.8.6 MeterMultiplierKind enumeration

Kind of meter multiplier.

Table 129 shows all literals of MeterMultiplierKind.

Table 129 – Literals of Metering::MeterMultiplierKind

literal	description
kH	Meter kh (watthour) constant. The number of watthours that must be applied to the meter to cause one disk revolution for an electromechanical meter or the number of watthours represented by one increment pulse for an electronic meter.
kR	Register multiplier. The number to multiply the register reading by in order to get kWh.
kE	Test constant.
ctRatio	Current transformer ratio used to convert associated quantities to real measurements.
ptRatio	Potential transformer ratio used to convert associated quantities to real measurements.
transformerRatio	Product of the CT ratio and PT ratio.

6.8.7 RandomisationKind enumeration

Kind of randomisation to be applied to control the timing of end device control commands and/or the definition of demand response and load control events. Value other than 'none' is typically used to mitigate potential deleterious effects of simultaneous operation of multiple devices.

Table 130 shows all literals of RandomisationKind.

Table 130 – Literals of Metering::RandomisationKind

literal	description
start	Start time of an event or control action affecting one or more multiple devices is randomised.
end	End time of an event or control action affecting one or more devices is randomised to prevent simultaneous operation.
startAndEnd	Both the start time and the end time of an event or control action affecting one or more devices are randomised to prevent simultaneous operation.
default	Randomisation of start and/or end times involving the operation of one or more devices is controlled by default settings for the device(s).
none	Neither the start time nor the end time of an event or control action affecting one or more devices is randomised.

6.8.8 ReadingReasonKind enumeration

Reason for the reading being taken.

Table 131 shows all literals of ReadingReasonKind.

Table 131 – Literals of Metering::ReadingReasonKind

literal	description
installation	Reading(s) taken or to be taken in conjunction with installation of a meter.
removal	Reading(s) taken or to be taken in conjunction with removal of a meter.
inquiry	Reading(s) taken or to be taken in response to an inquiry by a customer or other party.
billing	Reading(s) taken or to be taken in response to a billing-related inquiry by a customer or other party. A variant of 'inquiry'.
moveIn	Reading(s) taken or to be taken in conjunction with a customer move-in event.
moveOut	Reading(s) taken or to be taken in conjunction with a customer move-out event.
demandReset	Reading(s) taken or to be taken in conjunction with the resetting of one or more demand registers in a meter.
serviceDisconnect	Reading(s) taken or to be taken in conjunction with a disconnection of service.
serviceConnect	Reading(s) taken or to be taken in conjunction with a connection or re-connection of service.
loadManagement	Reading(s) taken or to be taken to support management of loads on distribution networks or devices.
loadResearch	Reading(s) taken or to be taken to support research and analysis of loads on distribution networks or devices.
other	Reading(s) taken or to be taken for some other reason or purpose.

6.8.9 ServiceMultiplierKind enumeration

Kind of service multiplier.

Table 132 shows all literals of ServiceMultiplierKind.

Table 132 – Literals of Metering::ServiceMultiplierKind

literal	description
ctRatio	Current transformer ratio used to convert associated quantities to real measurements.
ptRatio	Voltage transformer ratio used to convert associated quantities to real measurements.
transformerRatio	Product of the CT ratio and PT ratio.

6.8.10 TransmissionModeKind enumeration

Transmission mode for end device display controls, applicable to premises area network (PAN) devices.

Table 133 shows all literals of TransmissionModeKind.

Table 133 – Literals of Metering::TransmissionModeKind

literal	description
normal	Message transmission mode whereby messages or commands are sent to specific devices.
anonymous	Message transmission mode whereby messages or commands are broadcast to unspecified devices listening for such communications.
both	Message transmission mode whereby messages or commands are sent by both 'normal' and 'anonymous' methods.

6.8.11 UsagePointConnectedKind enumeration

State of the usage point with respect to connection to the network.

Table 134 shows all literals of UsagePointConnectedKind.

Table 134 – Literals of Metering::UsagePointConnectedKind

literal	description
connected	The usage point is connected to the network and able to receive or send the applicable commodity (electricity, gas, water, etc.).
physicallyDisconnected	The usage point has been disconnected from the network at a point upstream of the meter. The usage point is unable to receive or send the applicable commodity (electricity, gas, water, etc.). A physical disconnect is often achieved by utilising a field crew.
logicallyDisconnected	The usage point has been disconnected through operation of a disconnect function within the meter present at the usage point. The usage point is unable to receive or send the applicable commodity (electricity, gas, water, etc.) A logical disconnect can often be achieved without utilising a field crew.

6.8.12 ControlledAppliance compound

Appliance controlled with a PAN device control.

Table 135 shows all attributes of ControlledAppliance.

Table 135 – Attributes of Metering::ControlledAppliance

name	type	description
isElectricVehicle	Boolean	True if the appliance is an electric vehicle.
isExteriorLighting	Boolean	True if the appliance is exterior lighting.
isInteriorLighting	Boolean	True if the appliance is interior lighting.
isGenerationSystem	Boolean	True if the appliance is a generation system.
isHvacCompressorOrFurnace	Boolean	True if the appliance is HVAC compressor or furnace.
isIrrigationPump	Boolean	True if the appliance is an irrigation pump.
isManagedCommercialIndustrialLoad	Boolean	True if the appliance is managed commercial or industrial load.
isPoolPumpSpaJacuzzi	Boolean	True if the appliance is a pool, pump, spa or jacuzzi.
isSimpleMiscLoad	Boolean	True if the appliance is a simple miscellaneous load.
isSmartAppliance	Boolean	True if the appliance is a smart appliance.
isStripAndBaseboardHeater	Boolean	True if the appliance is a stip or baseboard heater.
isWaterHeater	Boolean	True if the appliance is a water heater.

6.8.13 EndDeviceCapability compound

Inherent capabilities of an end device (i.e., the functions it supports).

Table 136 shows all attributes of EndDeviceCapability.

Table 136 – Attributes of Metering::EndDeviceCapability

name	type	description
reverseFlow	Boolean	True if reverse flow function is supported.
demandResponse	Boolean	True if demand response function is supported.
metrology	Boolean	True if metrology function is supported.
outageHistory	Boolean	True if outage history function is supported.
relaysProgramming	Boolean	True if relays programming function is supported.
onRequestRead	Boolean	True if on request read function is supported.
autonomousDst	Boolean	True if autonomous DST (daylight saving time) function is supported.
communication	Boolean	True if communication function is supported.
connectDisconnect	Boolean	True if connect and disconnect function is supported.
electricMetering	Boolean	True if electric metering function is supported.
gasMetering	Boolean	True if gas metering function is supported.
waterMetering	Boolean	True if water metering function is supported.
textMessage	Boolean	True if the displaying of text messages is supported.
pricingInfo	Boolean	True if pricing information is supported.
pulseOutput	Boolean	True if device produces pulse outputs.
pressureCompensation	Boolean	True if device performs pressure compensation for metered quantities.
temperatureCompensation	Boolean	True if device performs temperature compensation for metered quantities.
superCompressibilityCompensation	Boolean	True if device performs super compressibility compensation for metered quantities.

6.8.14 EndDeviceTiming compound

Timing for the control actions of end devices.

Table 137 shows all attributes of EndDeviceTiming.

Table 137 – Attributes of Metering::EndDeviceTiming

name	type	description
duration	Minutes	Duration of the end device control action or the business event that is the subject of the end device control.
durationIndefinite	Boolean	True if 'duration' is indefinite.
interval	DateTimeInterval	Start and end time of an interval during which end device control actions are to be executed.
randomisation	RandomisationKind	Kind of randomisation to be applied to the end device control actions to be executed.

6.8.15 RationalNumber compound

Rational number = 'numerator' / 'denominator'.

Table 138 shows all attributes of RationalNumber.

Table 138 – Attributes of Metering::RationalNumber

name	type	description
numerator	Integer	Numerator.
denominator	Integer	Denominator. Value 1 indicates the number is a simple integer.

6.8.16 ReadingInterharmonic compound

Interharmonics are represented as a rational number 'numerator' / 'denominator', and harmonics are represented using the same mechanism and identified by 'denominator'=1.

Table 139 shows all attributes of ReadingInterharmonic.

Table 139 – Attributes of Metering::ReadingInterharmonic

name	type	description
numerator	Integer	Interharmonic numerator. Value 0 means not applicable. Value 1 is used in combination with 'denominator'=2 to represent interharmonic 1/2, and with 'denominator'=1 it represents fundamental frequency. Finally, values greater than 1 indicate the harmonic of that order (e.g., 'numerator'=5 is the fifth harmonic).
denominator	Integer	Interharmonic denominator. Value 0 means not applicable. Value 2 is used in combination with 'numerator'=1 to represent interharmonic 1/2. Finally, value 1 indicates the harmonic of the order specified with 'numerator'.

6.8.17 BaseReading

Common representation for reading values. Note that a reading value may have multiple qualities, as produced by various systems ('ReadingQuality.source').

Table 140 shows all attributes of BaseReading.

Table 140 – Attributes of Metering::BaseReading

name	type	description
value	String	Value of this reading.
source	String	System that originally supplied the reading (e.g., customer, AMI system, handheld reading system, another enterprise system, etc.).
timePeriod	DateTimeInterval	Start and end of the period for those readings whose type has a time attribute such as 'billing', seasonal' or 'forTheSpecifiedPeriod'.
reportedDateTime	DateTime	(used only when there are detailed auditing requirements) Date and time at which the reading was first delivered to the metering system.
timeStamp	DateTime	inherited from: MeasurementValue
sensorAccuracy	PerCent	inherited from: MeasurementValue
aliasName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject

Table 141 shows all association ends of BaseReading with other classes.

Table 141 – Association ends of Metering::BaseReading with other classes

[mult from]	[mult to] name	type	description
[0..1]	[0..*] ReadingQualities	ReadingQuality	All qualities of this reading.
[1..1]	[0..1] RemoteSource	RemoteSource	inherited from: MeasurementValue
[1..1]	[0..1] MeasurementValueQuality	MeasurementValueQuality	inherited from: MeasurementValue
[0..*]	[1..1] MeasurementValueSource	MeasurementValueSource	inherited from: MeasurementValue
[0..1]	[0..*] DiagramObjects	DiagramObject	inherited from: IdentifiedObject
[1..1]	[0..*] Names	Name	inherited from: IdentifiedObject

6.8.18 Channel

A single path for the collection or reporting of register values over a period of time. For example, a register which measures forward energy can have two channels, one providing bulk quantity readings and the other providing interval readings of a fixed interval size.

Table 142 shows all attributes of Channel.

Table 142 – Attributes of Metering::Channel

name	type	description
isVirtual	Boolean	If true, the data is being calculated by an enterprise system rather than metered directly.
aliasName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject

Table 143 shows all association ends of Channel with other classes.

Table 143 – Association ends of Metering::Channel with other classes

[mult from]	[mult to] name	type	description
[0..*]	[0..1] Register	Register	Register whose values are collected/reported by this channel.
[0..1]	[0..1] ReadingType	ReadingType	Reading type for register values reported/collected by this channel.
[0..1]	[0..*] DiagramObjects	DiagramObject	inherited from: IdentifiedObject
[1..1]	[0..*] Names	Name	inherited from: IdentifiedObject

6.8.19 ComFunction

Communication function of communication equipment or a device such as a meter.

Table 144 shows all attributes of ComFunction.

Table 144 – Attributes of Metering::ComFunction

name	type	description
amrAddress	String	Communication ID number (e.g. serial number, IP address, telephone number, etc.) of the AMR module which serves this meter.
amrRouter	String	Communication ID number (e.g. port number, serial number, data collector ID, etc.) of the parent device associated to this AMR module.
direction	ComDirectionKind	Kind of communication direction.
technology	ComTechnologyKind	Kind of communication technology.
enabled	Boolean	inherited from: EndDeviceFunction
programID	String	inherited from: AssetFunction
firmwareID	String	inherited from: AssetFunction
hardwareID	String	inherited from: AssetFunction
password	String	inherited from: AssetFunction
configID	String	inherited from: AssetFunction
aliasName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject

Table 145 shows all association ends of ComFunction with other classes.

Table 145 – Association ends of Metering::ComFunction with other classes

[mult from]	[mult to] name	type	description
[0..*]	[0..1] ComModule	ComModule	Module performing this communication function.
[0..*]	[0..1] EndDevice	EndDevice	inherited from: EndDeviceFunction
[0..1]	[0..*] Registers	Register	inherited from: EndDeviceFunction
[0..1]	[0..*] DiagramObjects	DiagramObject	inherited from: IdentifiedObject
[1..1]	[0..*] Names	Name	inherited from: IdentifiedObject

6.8.20 ComModule

An asset having communications capabilities that can be paired with a meter or other end device to provide the device with communication ability, through associated communication function. An end device that has communications capabilities through embedded hardware can use that function directly (without the communication module), or combine embedded communication function with additional communication functions provided through an external communication module (e.g. zigbee).

Table 146 shows all attributes of ComModule.

Table 146 – Attributes of Metering::ComModule

name	type	description
amrSystem	String	Automated meter reading (AMR) system communicating with this com module.
timeZoneOffset	Minutes	Time zone offset relative to GMT for the location of this com module.
supportsAutonomousDst	Boolean	If true, autonomous DST (daylight savings time) function is supported.
type	String	inherited from: Asset
utcNumber	String	inherited from: Asset
serialNumber	String	inherited from: Asset
lotNumber	String	inherited from: Asset
purchasePrice	Money	inherited from: Asset
critical	Boolean	inherited from: Asset
electronicAddress	ElectronicAddress	inherited from: Asset
lifecycle	LifecycleDate	inherited from: Asset
acceptanceTest	AcceptanceTest	inherited from: Asset
initialCondition	String	inherited from: Asset
initialLossOfLife	PerCent	inherited from: Asset
status	Status	inherited from: Asset
aliasName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject

Table 147 shows all association ends of ComModule with other classes.

Table 147 – Association ends of Metering::ComModule with other classes

[mult from]	[mult to] name	type	description
[0..1]	[0..*] ComFunctions	ComFunction	All functions this communication module performs.
[0..*]	[0..*] PowerSystemResources	PowerSystemResource	inherited from: Asset
[0..*]	[0..*] ActivityRecords	ActivityRecord	inherited from: Asset
[0..1]	[0..*] ConfigurationEvents	ConfigurationEvent	inherited from: Asset
[0..*]	[0..1] Location	Location	inherited from: Asset
[0..*]	[0..*] OrganisationRoles	AssetOrganisationRole	inherited from: Asset
[0..*]	[0..1] AssetContainer	AssetContainer	inherited from: Asset
[0..*]	[0..1] AssetInfo	AssetInfo	inherited from: Asset

[mult from]	[mult to] name	type	description
[0..1]	[0..*] DiagramObjects	DiagramObject	inherited from: IdentifiedObject
[1..1]	[0..*] Names	Name	inherited from: IdentifiedObject

6.8.21 DemandResponseProgram

Demand response program.

Table 148 shows all attributes of DemandResponseProgram.

Table 148 – Attributes of Metering::DemandResponseProgram

name	type	description
type	String	Type of demand response program; examples are CPP (critical-peak pricing), RTP (real-time pricing), DLC (direct load control), DBP (demand bidding program), BIP (base interruptible program). Note that possible types change a lot and it would be impossible to enumerate them all.
validityInterval	DateTimeInterval	Interval within which the program is valid.
aliasName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject

Table 149 shows all association ends of DemandResponseProgram with other classes.

Table 149 – Association ends of Metering::DemandResponseProgram with other classes

[mult from]	[mult to] name	type	description
[0..*]	[0..*] CustomerAgreements	CustomerAgreement	All customer agreements through which the customer is enrolled in this demand response program.
[0..*]	[0..*] UsagePointGroups	UsagePointGroup	All usage point groups enrolled in this demand response program.
[0..*]	[0..*] EndDeviceGroups	EndDeviceGroup	All groups of end devices enrolled in this demand response program.
[0..1]	[0..*] DiagramObjects	DiagramObject	inherited from: IdentifiedObject
[1..1]	[0..*] Names	Name	inherited from: IdentifiedObject

6.8.22 EndDevice

Asset container that performs one or more end device functions. One type of end device is a meter which can perform metering, load management, connect/disconnect, accounting functions, etc. Some end devices, such as ones monitoring and controlling air conditioners, refrigerators, pool pumps may be connected to a meter. All end devices may have communication capability defined by the associated communication function(s). An end device may be owned by a consumer, a service provider, utility or otherwise.

There may be a related end device function that identifies a sensor or control point within a metering application or communications systems (e.g., water, gas, electricity).

Some devices may use an optical port that conforms to the ANSI C12.18 standard for communications.

Table 150 shows all attributes of EndDevice.

Table 150 – Attributes of Metering::EndDevice

name	type	description
isVirtual	Boolean	If true, there is no physical device. As an example, a virtual meter can be defined to aggregate the consumption for two or more physical meters. Otherwise, this is a physical hardware device.
isPan	Boolean	If true, this is a premises area network (PAN) device.
installCode	String	Installation code.
amrSystem	String	Automated meter reading (AMR) system responsible for communications to this end device.
timeZoneOffset	Minutes	Time zone offset relative to GMT for the location of this end device.
type	String	inherited from: Asset
utcNumber	String	inherited from: Asset
serialNumber	String	inherited from: Asset
lotNumber	String	inherited from: Asset
purchasePrice	Money	inherited from: Asset
critical	Boolean	inherited from: Asset
electronicAddress	ElectronicAddress	inherited from: Asset
lifecycle	LifecycleDate	inherited from: Asset
acceptanceTest	AcceptanceTest	inherited from: Asset
initialCondition	String	inherited from: Asset
initialLossOfLife	PerCent	inherited from: Asset
status	Status	inherited from: Asset
aliasName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject

Table 151 shows all association ends of EndDevice with other classes.

Table 151 – Association ends of Metering::EndDevice with other classes

[mult from]	[mult to] name	type	description
[0..*]	[0..1] EndDeviceInfo	EndDeviceInfo	End device data.
[0..1]	[0..*] EndDeviceFunctions	EndDeviceFunction	All end device functions this end device performs.
[0..*]	[0..1] Customer	Customer	Customer owning this end device.
[0..*]	[0..1] ServiceLocation	ServiceLocation	Service location whose service delivery is measured by this end device.
[0..*]	[0..*] EndDeviceGroups	EndDeviceGroup	All end device groups referring to this end device.
[0..*]	[0..*] EndDeviceControls	EndDeviceControl	All end device controls sending commands to this end device.
[0..1]	[0..*] EndDeviceEvents	EndDeviceEvent	All events reported by this end device.
[0..*]	[0..1] UsagePoint	UsagePoint	Usage point to which this end device belongs.
[0..1]	[0..*] Seals	Seal	inherited from: AssetContainer

[mult from]	[mult to] name	type	description
[0..1]	[0..*] Assets	Asset	inherited from: AssetContainer
[0..*]	[0..*] PowerSystemResources	PowerSystemResource	inherited from: Asset
[0..*]	[0..*] ActivityRecords	ActivityRecord	inherited from: Asset
[0..1]	[0..*] ConfigurationEvents	ConfigurationEvent	inherited from: Asset
[0..*]	[0..1] Location	Location	inherited from: Asset
[0..*]	[0..*] OrganisationRoles	AssetOrganisationRole	inherited from: Asset
[0..*]	[0..1] AssetContainer	AssetContainer	inherited from: Asset
[0..*]	[0..1] AssetInfo	AssetInfo	inherited from: Asset
[0..1]	[0..*] DiagramObjects	DiagramObject	inherited from: IdentifiedObject
[1..1]	[0..*] Names	Name	inherited from: IdentifiedObject

6.8.23 EndDeviceAction root class

Action/command performed by an end device on a device other than the end device.

Table 152 shows all attributes of EndDeviceAction.

Table 152 – Attributes of Metering::EndDeviceAction

name	type	description
command	String	Command text.
duration	Minutes	Amount of time the action of this control is to remain active.
durationIndefinite	Boolean	True if the action of this control is indefinite.
startDateTime	DateTime	Start date and time for action of this control.

Table 153 shows all association ends of EndDeviceAction with other classes.

Table 153 – Association ends of Metering::EndDeviceAction with other classes

[mult from]	[mult to] name	type	description
[0..1]	[0..1] EndDeviceControl	EndDeviceControl	End device control issuing this end device action.

6.8.24 EndDeviceControl

Instructs an end device (or an end device group) to perform a specified action.

Table 154 shows all attributes of EndDeviceControl.

Table 154 – Attributes of Metering::EndDeviceControl

name	type	description
issuerTrackingID	String	Identifier assigned by the initiator (e.g. retail electric provider) of an end device control action to uniquely identify the demand response event, text message, or other subject of the control action. Can be used when cancelling an event or text message request or to identify the originating event or text message in a consequential end device event.
issuerID	String	Unique identifier of the business entity originating an end device control.
reason	String	Reason for the control action that allows to determine how to continue processing. For example, disconnect meter command may require different processing by the receiving system if it has been issued for a network-related reason (protection) or for a payment-related reason.
scheduledInterval	DateTimeInterval	(if control has scheduled duration) Date and time interval the control has been scheduled to execute within.
priceSignal	FloatQuantity	(if applicable) Price signal used as parameter for this end device control.
drProgramLevel	Integer	Level of a demand response program request, where 0=emergency. Note: Attribute is not defined on DemandResponseProgram as it is not its inherent property (it serves to control it).
drProgramMandatory	Boolean	Whether a demand response program request is mandatory. Note: Attribute is not defined on DemandResponseProgram as it is not its inherent property (it serves to control it).
primaryDeviceTiming	EndDeviceTiming	Timing for the control actions performed on the device identified in the end device control.
secondaryDeviceTiming	EndDeviceTiming	Timing for the control actions performed by devices that are responding to event related information sent to the primary device indicated in the end device control. For example, load control actions performed by a PAN device in response to demand response event information sent to a PAN gateway server.
aliasName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject

Table 155 shows all association ends of EndDeviceControl with other classes.

Table 155 – Association ends of Metering::EndDeviceControl with other classes

[mult from]	[mult to] name	type	description
[0..*]	[0..*] EndDevices	EndDevice	All end devices receiving commands from this end device control.
[0..1]	[0..1] EndDeviceAction	EndDeviceAction	End device action issued by this end device control.
[0..*]	[0..*] UsagePointGroups	UsagePointGroup	All usage point groups receiving commands from this end device control.
[0..*]	[1..1] EndDeviceControlType	EndDeviceControlType	Type of this end device control.
[0..*]	[0..*] UsagePoints	UsagePoint	All usage points receiving commands from this end device control.
[0..*]	[0..*] EndDeviceGroups	EndDeviceGroup	All end device groups receiving commands from this end device control.
[0..1]	[0..*] DiagramObjects	DiagramObject	inherited from: IdentifiedObject
[1..1]	[0..*] Names	Name	inherited from: IdentifiedObject

6.8.25 EndDeviceControlType

Detailed description for a control produced by an end device. Values in attributes allow for creation of recommended codes to be used for identifying end device controls as follows: <type>.<domain>.<subDomain>.<eventOrAction>.

Table 156 shows all attributes of EndDeviceControlType.

Table 156 – Attributes of Metering::EndDeviceControlType

name	type	description
type	String	Type of physical device from which the control was created. A value of zero (0) can be used when the source is unknown.
domain	String	High-level nature of the control.
subDomain	String	More specific nature of the control, as a further sub-categorisation of 'domain'.
eventOrAction	String	The most specific part of this control type. It is mainly in the form of a verb that gives action to the control that just occurred.
aliasName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject

Table 157 shows all association ends of EndDeviceControlType with other classes.

Table 157 – Association ends of Metering::EndDeviceControlType with other classes

[mult from]	[mult to] name	type	description
[1..1]	[0..*] EndDeviceControls	EndDeviceControl	All end device controls of this type.
[0..1]	[0..*] DiagramObjects	DiagramObject	inherited from: IdentifiedObject
[1..1]	[0..*] Names	Name	inherited from: IdentifiedObject

6.8.26 EndDeviceEvent

Event detected by a device function associated with the end device.

Table 158 shows all attributes of EndDeviceEvent.

Table 158 – Attributes of Metering::EndDeviceEvent

name	type	description
userID	String	(if user initiated) ID of user who initiated this end device event.
issuerTrackingID	String	Identifier assigned by the initiator (e.g. retail electric provider) of an end device control action to uniquely identify the demand response event, text message, or other subject of the control action. Can be used when cancelling an event or text message request or to identify the originating event or text message in a consequential end device event.
issuerID	String	Unique identifier of the business entity originating an end device control.
createdDateTime	DateTime	inherited from: ActivityRecord
type	String	inherited from: ActivityRecord
severity	String	inherited from: ActivityRecord
reason	String	inherited from: ActivityRecord
status	Status	inherited from: ActivityRecord
aliasName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject

Table 159 shows all association ends of EndDeviceEvent with other classes.

Table 159 – Association ends of Metering::EndDeviceEvent with other classes

[mult from]	[mult to] name	type	description
[0..*]	[0..1] EndDevice	EndDevice	End device that reported this end device event.
[0..*]	[1..1] EndDeviceEventType	EndDeviceEventType	Type of this end device event.
[0..1]	[0..*] EndDeviceEventDetails	EndDeviceEventDetail	All details of this end device event.
[0..*]	[0..1] UsagePoint	UsagePoint	Usage point for which this end device event is reported.
[0..*]	[0..1] MeterReading	MeterReading	Set of measured values to which this event applies.
[0..*]	[0..*] Assets	Asset	inherited from: ActivityRecord
[0..1]	[0..*] DiagramObjects	DiagramObject	inherited from: IdentifiedObject
[1..1]	[0..*] Names	Name	inherited from: IdentifiedObject

6.8.27 EndDeviceEventDetail root class

Name-value pair, specific to end device events.

Table 160 shows all attributes of EndDeviceEventDetail.

Table 160 – Attributes of Metering::EndDeviceEventDetail

name	type	description
name	String	Name.
value	StringQuantity	Value, including unit information.

Table 161 shows all association ends of EndDeviceEventDetail with other classes.

Table 161 – Association ends of Metering::EndDeviceEventDetail with other classes

[mult from]	[mult to] name	type	description
[0..*]	[0..1] EndDeviceEvent	EndDeviceEvent	End device owning this detail.

6.8.28 EndDeviceEventType

Detailed description for an event produced by an end device. Values in attributes allow for creation of recommended codes to be used for identifying end device events as follows: <type>.<domain>.<subDomain>.<eventOrAction>.

Table 162 shows all attributes of EndDeviceEventType.

Table 162 – Attributes of Metering::EndDeviceEventType

name	type	description
type	String	Type of physical device from which the event was created. A value of zero (0) can be used when the source is unknown.
domain	String	High-level nature of the event. By properly classifying events by a small set of domain codes, a system can more easily run reports based on the types of events that have occurred or been received.
subDomain	String	More specific nature of the event, as a further sub-categorisation of 'domain'.
eventOrAction	String	The most specific part of this event type. It is mainly in the form of a verb that gives action to the event that just occurred.
aliasName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject

Table 163 shows all association ends of EndDeviceEventType with other classes.

Table 163 – Association ends of Metering::EndDeviceEventType with other classes

[mult from]	[mult to] name	type	description
[1..1]	[0..*] EndDeviceEvents	EndDeviceEvent	All end device events of this type.
[0..1]	[0..*] DiagramObjects	DiagramObject	inherited from: IdentifiedObject
[1..1]	[0..*] Names	Name	inherited from: IdentifiedObject

6.8.29 EndDeviceFunction

Function performed by an end device such as a meter, communication equipment, controllers, etc.

Table 164 shows all attributes of EndDeviceFunction.

Table 164 – Attributes of Metering::EndDeviceFunction

name	type	description
enabled	Boolean	True if the function is enabled.
programID	String	inherited from: AssetFunction
firmwareID	String	inherited from: AssetFunction
hardwareID	String	inherited from: AssetFunction
password	String	inherited from: AssetFunction
configID	String	inherited from: AssetFunction
aliasName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject

Table 165 shows all association ends of EndDeviceFunction with other classes.

Table 165 – Association ends of Metering::EndDeviceFunction with other classes

[mult from]	[mult to] name	type	description
[0..*]	[0..1] EndDevice	EndDevice	End device that performs this function.
[0..1]	[0..*] Registers	Register	All registers for quantities metered by this end device function.
[0..1]	[0..*] DiagramObjects	DiagramObject	inherited from: IdentifiedObject
[1..1]	[0..*] Names	Name	inherited from: IdentifiedObject

6.8.30 EndDeviceGroup

Abstraction for management of group communications within a two-way AMR system or the data for a group of related end devices. Commands can be issued to all of the end devices that belong to the group using a defined group address and the underlying AMR communication infrastructure.

Table 166 shows all attributes of EndDeviceGroup.

Table 166 – Attributes of Metering::EndDeviceGroup

name	type	description
aliasName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject

Table 167 shows all association ends of EndDeviceGroup with other classes.

Table 167 – Association ends of Metering::EndDeviceGroup with other classes

[mult from]	[mult to] name	type	description
[0..*]	[0..*] EndDevices	EndDevice	All end devices this end device group refers to.
[0..*]	[0..*] EndDeviceControls	EndDeviceControl	All end device controls sending commands to this end device group.
[0..*]	[0..*] DemandResponsePrograms	DemandResponseProgram	All demand response programs this group of end devices is enrolled in.
[0..1]	[0..*] DiagramObjects	DiagramObject	inherited from: IdentifiedObject
[1..1]	[0..*] Names	Name	inherited from: IdentifiedObject

6.8.31 EndDeviceInfo

End device data.

Table 168 shows all attributes of EndDeviceInfo.

Table 168 – Attributes of Metering::EndDeviceInfo

name	type	description
capability	EndDeviceCapability	Inherent capabilities of the device (i.e., the functions it supports).
phaseCount	Integer	Number of potential phases the end device supports, typically 0, 1 or 3.
ratedCurrent	CurrentFlow	Rated current.
ratedVoltage	Voltage	Rated voltage.
isSolidState	Boolean	If true, this is a solid state end device (as opposed to a mechanical or electromechanical device).
aliasName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject

Table 169 shows all association ends of EndDeviceInfo with other classes.

Table 169 – Association ends of Metering::EndDeviceInfo with other classes

[mult from]	[mult to] name	type	description
[0..1]	[0..*] EndDevices	EndDevice	All end devices described with this data.
[0..1]	[0..*] PowerSystemResource	PowerSystemResource	inherited from: AssetInfo
[0..1]	[0..*] Assets	Asset	inherited from: AssetInfo
[0..1]	[0..1] AssetModel	AssetModel	inherited from: AssetInfo
[0..1]	[0..*] DiagramObjects	DiagramObject	inherited from: IdentifiedObject
[1..1]	[0..*] Names	Name	inherited from: IdentifiedObject

6.8.32 IntervalBlock root class

Time sequence of readings of the same reading type. Contained interval readings may need conversion through the application of an offset and a scalar defined in associated pending.

Table 170 shows all association ends of IntervalBlock with other classes.

Table 170 – Association ends of Metering::IntervalBlock with other classes

[mult from]	[mult to] name	type	description
[0..*]	[0..1] PendingCalculation	PendingCalculation	Pending calculation to apply to interval reading values contained by this block (after which the resulting reading type is different than the original because it reflects the conversion result).
[0..*]	[0..*] IntervalReadings	IntervalReading	Interval reading contained in this block.
[0..*]	[1..1] ReadingType	ReadingType	Type information for interval reading values contained in this block.
[0..*]	[0..1] MeterReading	MeterReading	Meter reading containing this interval block.

6.8.33 IntervalReading

Data captured at regular intervals of time. Interval data could be captured as incremental data, absolute data, or relative data. The source for the data is usually a tariff quantity or an engineering quantity. Data is typically captured in time-tagged, uniform, fixed-length intervals of 5 min, 10 min, 15 min, 30 min, or 60 min.

NOTE Interval Data is sometimes also called "Interval Data Readings" (IDR).

Table 171 shows all attributes of IntervalReading.

Table 171 – Attributes of Metering::IntervalReading

name	type	description
value	Float	inherited from: BaseReading
source	String	inherited from: BaseReading
timePeriod	DateTimeInterval	inherited from: BaseReading
reportedDateTime	DateTime	inherited from: BaseReading
timeStamp	DateTime	inherited from: MeasurementValue
sensorAccuracy	PerCent	inherited from: MeasurementValue
aliasName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject

Table 172 shows all association ends of IntervalReading with other classes.

Table 172 – Association ends of Metering::IntervalReading with other classes

[mult from]	[mult to] name	type	description
[0..*]	[0..*] IntervalBlocks	IntervalBlock	All blocks containing this interval reading.
[0..1]	[0..*] ReadingQualities	ReadingQuality	inherited from: BaseReading
[1..1]	[0..1] RemoteSource	RemoteSource	inherited from: MeasurementValue
[1..1]	[0..1] MeasurementValueQuality	MeasurementValueQuality	inherited from: MeasurementValue

[mult from]	[mult to] name	type	description
[0..*]	[1..1] MeasurementValueSource	MeasurementValueSource	inherited from: MeasurementValue
[0..1]	[0..*] DiagramObjects	DiagramObject	inherited from: IdentifiedObject
[1..1]	[0..*] Names	Name	inherited from: IdentifiedObject

6.8.34 Meter

Physical asset that performs the metering role of the usage point. Used for measuring consumption and detection of events.

Table 173 shows all attributes of Meter.

Table 173 – Attributes of Metering::Meter

name	type	description
formNumber	String	Meter form designation per ANSI C12.10 or other applicable standard. An alphanumeric designation denoting the circuit arrangement for which the meter is applicable and its specific terminal arrangement.
isVirtual	Boolean	inherited from: EndDevice
isPan	Boolean	inherited from: EndDevice
installCode	String	inherited from: EndDevice
amrSystem	String	inherited from: EndDevice
timeZoneOffset	Minutes	inherited from: EndDevice
type	String	inherited from: Asset
utcNumber	String	inherited from: Asset
serialNumber	String	inherited from: Asset
lotNumber	String	inherited from: Asset
purchasePrice	Money	inherited from: Asset
critical	Boolean	inherited from: Asset
electronicAddress	ElectronicAddress	inherited from: Asset
lifecycle	LifecycleDate	inherited from: Asset
acceptanceTest	AcceptanceTest	inherited from: Asset
initialCondition	String	inherited from: Asset
initialLossOfLife	PerCent	inherited from: Asset
status	Status	inherited from: Asset
aliasName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject

Table 174 shows all association ends of Meter with other classes.

Table 174 – Association ends of Metering::Meter with other classes

[mult from]	[mult to] name	type	description
[0..1]	[0..*] MeterMultipliers	MeterMultiplier	All multipliers applied at this meter.
[0..1]	[0..*] MeterReadings	MeterReading	All meter readings provided by this meter.
[0..1]	[0..*] MeterServiceWorks	MeterServiceWork	All non-replacement works on this meter.
[0..1]	[0..*] VendingTransactions	Transaction	All vending transactions on this meter.
[0..1]	[0..*] MeterReplacementWorks	MeterServiceWork	All works on replacement of this old meter.
[0..*]	[0..1] EndDeviceInfo	EndDeviceInfo	inherited from: EndDevice
[0..1]	[0..*] EndDeviceFunctions	EndDeviceFunction	inherited from: EndDevice
[0..*]	[0..1] Customer	Customer	inherited from: EndDevice
[0..*]	[0..1] ServiceLocation	ServiceLocation	inherited from: EndDevice
[0..*]	[0..*] EndDeviceGroups	EndDeviceGroup	inherited from: EndDevice
[0..*]	[0..*] EndDeviceControls	EndDeviceControl	inherited from: EndDevice
[0..1]	[0..*] EndDeviceEvents	EndDeviceEvent	inherited from: EndDevice
[0..*]	[0..1] UsagePoint	UsagePoint	inherited from: EndDevice
[0..1]	[0..*] Seals	Seal	inherited from: AssetContainer
[0..1]	[0..*] Assets	Asset	inherited from: AssetContainer
[0..*]	[0..*] PowerSystemResources	PowerSystemResource	inherited from: Asset
[0..*]	[0..*] ActivityRecords	ActivityRecord	inherited from: Asset
[0..1]	[0..*] ConfigurationEvents	ConfigurationEvent	inherited from: Asset
[0..*]	[0..1] Location	Location	inherited from: Asset
[0..*]	[0..*] OrganisationRoles	AssetOrganisationRole	inherited from: Asset
[0..*]	[0..1] AssetContainer	AssetContainer	inherited from: Asset
[0..*]	[0..1] AssetInfo	AssetInfo	inherited from: Asset
[0..1]	[0..*] DiagramObjects	DiagramObject	inherited from: IdentifiedObject
[1..1]	[0..*] Names	Name	inherited from: IdentifiedObject

6.8.35 MeterMultiplier

Multiplier applied at the meter.

Table 175 shows all attributes of MeterMultiplier.

Table 175 – Attributes of Metering::MeterMultiplier

name	type	description
kind	MeterMultiplierKind	Kind of multiplier.
value	Float	Multiplier value.
aliasName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject

Table 176 shows all association ends of MeterMultiplier with other classes.

Table 176 – Association ends of Metering::MeterMultiplier with other classes

[mult from]	[mult to] name	type	description
[0..*]	[0..1] Meter	Meter	Meter applying this multiplier.
[0..1]	[0..*] DiagramObjects	DiagramObject	inherited from: IdentifiedObject
[1..1]	[0..*] Names	Name	inherited from: IdentifiedObject

6.8.36 MeterReading

Set of values obtained from the meter.

Table 177 shows all attributes of MeterReading.

Table 177 – Attributes of Metering::MeterReading

name	type	description
valuesInterval	DateTimeInterval	Date and time interval of the data items contained within this meter reading.
isCoincidentTrigger	Boolean	If true, this meter reading is the meter reading for which other coincident meter readings are requested or provided.
aliasName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject

Table 178 shows all association ends of MeterReading with other classes.

Table 178 – Association ends of Metering::MeterReading with other classes

[mult from]	[mult to] name	type	description
[0..*]	[0..1] CustomerAgreement	CustomerAgreement	(could be deprecated in the future) Customer agreement for this meter reading.
[0..1]	[0..*] EndDeviceEvents	EndDeviceEvent	All end device events associated with this set of measured values.
[0..*]	[0..1] UsagePoint	UsagePoint	Usage point from which this meter reading (set of values) has been obtained.
[0..1]	[0..*] IntervalBlocks	IntervalBlock	All interval blocks contained in this meter reading.
[0..*]	[0..1] Meter	Meter	Meter providing this reading.
[0..*]	[0..*] Readings	Reading	All reading values contained within this meter reading.
[0..1]	[0..*] DiagramObjects	DiagramObject	inherited from: IdentifiedObject
[1..1]	[0..*] Names	Name	inherited from: IdentifiedObject

6.8.37 MeterServiceWork

Work involving meters.

Table 179 shows all attributes of MeterServiceWork.

Table 179 – Attributes of Metering::MeterServiceWork

name	type	description
kind	WorkKind	inherited from: Work
priority	String	inherited from: Work
requestDateTime	DateTime	inherited from: Work
type	String	inherited from: Document
createdDateTime	DateTime	inherited from: Document
lastModifiedDateTime	DateTime	inherited from: Document
revisionNumber	String	inherited from: Document
electronicAddress	ElectronicAddress	inherited from: Document
subject	String	inherited from: Document
title	String	inherited from: Document
docStatus	Status	inherited from: Document
status	Status	inherited from: Document
aliasName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject

Table 180 shows all association ends of MeterServiceWork with other classes.

Table 180 – Association ends of Metering::MeterServiceWork with other classes

[mult from]	[mult to] name	type	description
[0..*]	[0..1] Meter	Meter	Meter on which this non-replacement work is performed.
[0..*]	[0..1] OldMeter	Meter	Old meter replaced by this work.
[0..*]	[0..1] UsagePoint	UsagePoint	Usage point to which this meter service work applies.
[0..*]	[0..*] Customers	Customer	inherited from: Work
[0..1]	[0..*] ConfigurationEvents	ConfigurationEvent	inherited from: Document
[0..1]	[0..*] DiagramObjects	DiagramObject	inherited from: IdentifiedObject
[1..1]	[0..*] Names	Name	inherited from: IdentifiedObject

6.8.38 MetrologyRequirement

A specification of the metering requirements for a particular point within a network.

Table 181 shows all attributes of MetrologyRequirement.

Table 181 – Attributes of Metering::MetrologyRequirement

name	type	description
reason	ReadingReasonKind	Reason for this metrology requirement being specified.
aliasName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject

Table 182 shows all association ends of MetrologyRequirement with other classes.

Table 182 – Association ends of Metering::MetrologyRequirement with other classes

[mult from]	[mult to] name	type	description
[0..*]	[0..*] UsagePoints	UsagePoint	All usage points having this metrology requirement.
[0..*]	[1..*] ReadingTypes	ReadingType	All reading types required to be collected by this metrology requirement.
[0..1]	[0..*] DiagramObjects	DiagramObject	inherited from: IdentifiedObject
[1..1]	[0..*] Names	Name	inherited from: IdentifiedObject

6.8.39 PanDemandResponse

PAN control used to issue action/command to PAN devices during a demand response/load control event.

Table 183 shows all attributes of PanDemandResponse.

Table 183 – Attributes of Metering::PanDemandResponse

name	type	description
appliance	ControlledAppliance	Appliance being controlled.
avgLoadAdjustment	PerCent	Used to define a maximum energy usage limit as a percentage of the client implementations specific average energy usage. The load adjustment percentage is added to 100 % creating a percentage limit applied to the client implementations specific average energy usage. A
coolingOffset	Temperature	Requested offset to apply to the normal cooling setpoint at the time of the start of the event. It represents a temperature change that will be applied to the associated cooling set point. The temperature offsets will be calculated per the local temperature in the thermostat. The calculated temperature will be interpreted as the number of degrees to be added to the cooling set point. Sequential demand response events are not cumulative. The offset shall be applied to the normal setpoint.
coolingSetpoint	Temperature	Requested cooling set point. Temperature set point is typically defined and calculated based on local temperature.
dutyCycle	PerCent	Maximum "on" state duty cycle as a percentage of time. For example, if the value is 80, the device would be in an "on" state for 80 % of the time for the duration of the action.
heatingOffset	Temperature	Requested offset to apply to the normal heating setpoint at the time of the start of the event. It represents a temperature change that will be applied to the associated heating set point. The temperature offsets will be calculated per the local temperature in the thermostat. The calculated temperature will be interpreted as the number of degrees to be subtracted from the heating set point. Sequential demand response events are not cumulative. The offset shall be applied to the normal setpoint.
heatingSetpoint	Temperature	Requested heating set point. Temperature set point is typically defined and calculated based on local temperature.

name	type	description
cancelControlMode	String	Encoding of cancel control.
cancelDateTime	DateTime	Timestamp when a canceling of the event is scheduled to start.
cancelNow	Boolean	If true, a canceling of the event should start immediately.
criticalityLevel	String	Level of criticality for the action of this control. The action taken by load control devices for an event can be solely based on this value, or in combination with other load control event fields supported by the device.
enrollmentGroup	String	Provides a mechanism to direct load control actions to groups of PAN devices. It can be used in conjunction with the PAN device types.
command	String	inherited from: EndDeviceAction
duration	Minutes	inherited from: EndDeviceAction
durationIndefinite	Boolean	inherited from: EndDeviceAction
startDateTime	DateTime	inherited from: EndDeviceAction

Table 184 shows all association ends of PanDemandResponse with other classes.

Table 184 – Association ends of Metering::PanDemandResponse with other classes

[mult from]	[mult to] name	type	description
[0..1]	[0..1] EndDeviceControl	EndDeviceControl	inherited from: EndDeviceAction

6.8.40 PanDisplay

PAN action/command used to issue the displaying of text messages on PAN devices.

Table 185 shows all attributes of PanDisplay.

Table 185 – Attributes of Metering::PanDisplay

name	type	description
confirmationRequired	Boolean	If true, the requesting entity (e.g. retail electric provider) requires confirmation of the successful display of the text message.
priority	String	Priority associated with the text message to be displayed.
textMessage	String	Text to be displayed by a PAN device.
transmissionMode	TransmissionModeKind	Transmission mode to be used for this PAN display control.
command	String	inherited from: EndDeviceAction
duration	Minutes	inherited from: EndDeviceAction
durationIndefinite	Boolean	inherited from: EndDeviceAction
startDateTime	DateTime	inherited from: EndDeviceAction

Table 186 shows all association ends of PanDisplay with other classes.

Table 186 – Association ends of Metering::PanDisplay with other classes

[mult from]	[mult to] name	type	description
[0..1]	[0..1] EndDeviceControl	EndDeviceControl	inherited from: EndDeviceAction

6.8.41 PanPricing

PAN action/command used to issue pricing information to a PAN device.

Table 187 shows all attributes of PanPricing.

Table 187 – Attributes of Metering::PanPricing

name	type	description
providerID	Integer	Unique identifier for the commodity provider.
command	String	inherited from: EndDeviceAction
duration	Minutes	inherited from: EndDeviceAction
durationIndefinite	Boolean	inherited from: EndDeviceAction
startDateTime	DateTime	inherited from: EndDeviceAction

Table 188 shows all association ends of PanPricing with other classes.

Table 188 – Association ends of Metering::PanPricing with other classes

[mult from]	[mult to] name	type	description
[0..1]	[0..*] PanPricingDetails	PanPricingDetail	All pricing details issued by this PAN pricing command/action.
[0..1]	[0..1] EndDeviceControl	EndDeviceControl	inherited from: EndDeviceAction

6.8.42 PanPricingDetail root class

Detail for a single price command/action.

Table 189 shows all attributes of PanPricingDetail.

Table 189 – Attributes of Metering::PanPricingDetail

name	type	description
alternateCostDelivered	Float	Alternative measure of the cost of the energy consumed. An example might be the emissions of CO_2 for each kWh of electricity consumed providing a measure of the environmental cost.
alternateCostUnit	String	Cost unit for the alternate cost delivered field. One example is kg of CO_2 per unit of measure.
currentTimeDate	DateTime	Current time as determined by a PAN device.
generationPrice	Money	Price of the commodity measured in base unit of currency per 'unitOfMeasure'.
generationPriceRatio	Float	Ratio of 'generationPrice' to the "normal" price chosen by the commodity provider.
priceTierCount	Integer	Maximum number of price tiers available.

name	type	description
price	Money	Price of the commodity measured in base unit of currency per 'unitOfMeasure'.
priceRatio	Float	Ratio of 'price' to the "normal" price chosen by the commodity provider.
priceTier	Integer	Pricing tier as chosen by the commodity provider.
rateLabel	String	Label of the current billing rate specified by commodity provider.
registerTier	String	Register tier accumulating usage information.
unitOfMeasure	String	Defines commodity as well as its base unit of measure.
priceTierLabel	String	Label for price tier.

Table 190 shows all association ends of PanPricingDetail with other classes.

Table 190 – Association ends of Metering::PanPricingDetail with other classes

[mult from]	[mult to] name	type	description
[0..*]	[0..1] PanPricing	PanPricing	PAN pricing command/action issuing this price detail.

6.8.43 PendingCalculation root class

When present, a scalar conversion that needs to be applied to every IntervalReading.value contained in IntervalBlock. This conversion results in a new associated ReadingType, reflecting the true dimensions of IntervalReading values after the conversion.

Table 191 shows all attributes of PendingCalculation.

Table 191 – Attributes of Metering::PendingCalculation

name	type	description
scalarFloat	Float	(if scalar is floating number) When multiplied with 'IntervalReading.value', it causes a unit of measure conversion to occur, according to the 'ReadingType.unit'.
scalarNumerator	Integer	(if scalar is integer or rational number) When the scalar is a simple integer, and this attribute is presented alone and multiplied with 'IntervalReading.value', it causes a unit of measure conversion to occur, resulting in the 'ReadingType.unit'. It is never used in conjunction with 'scalarFloat', only with 'scalarDenominator'.
scalarDenominator	Integer	(if scalar is rational number) When 'IntervalReading.value' is multiplied by 'scalarNumerator' and divided by this value, it causes a unit of measure conversion to occur, resulting in the 'ReadingType.unit'.
offset	Integer	(if applicable) Offset to be added as well as multiplication using scalars.
multiplyBeforeAdd	Boolean	Whether scalars should be applied before adding the 'offset'.

Table 192 shows all association ends of PendingCalculation with other classes.

Table 192 – Association ends of Metering::PendingCalculation with other classes

[mult from]	[mult to] name	type	description
[0..1]	[1..1] ReadingType	ReadingType	Reading type resulting from this pending conversion.
[0..1]	[0..*] IntervalBlocks	IntervalBlock	All blocks of interval reading values to which this pending conversion applies.

6.8.44 Reading

Specific value measured by a meter or other asset, or calculated by a system. Each Reading is associated with a specific ReadingType.

Table 193 shows all attributes of Reading.

Table 193 – Attributes of Metering::Reading

name	type	description
reason	ReadingReasonKind	Reason for this reading being taken.
value	Float	inherited from: BaseReading
source	String	inherited from: BaseReading
timePeriod	DateTimeInterval	inherited from: BaseReading
reportedDateTime	DateTime	inherited from: BaseReading
timeStamp	DateTime	inherited from: MeasurementValue
sensorAccuracy	PerCent	inherited from: MeasurementValue
aliasName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject

Table 194 shows all association ends of Reading with other classes.

Table 194 – Association ends of Metering::Reading with other classes

[mult from]	[mult to] name	type	description
[0..*]	[0..*] MeterReadings	MeterReading	All meter readings (sets of values) containing this reading value.
[0..*]	[1..1] ReadingType	ReadingType	Type information for this reading value.
[0..1]	[0..*] ReadingQualities	ReadingQuality	inherited from: BaseReading
[1..1]	[0..1] RemoteSource	RemoteSource	inherited from: MeasurementValue
[1..1]	[0..1] MeasurementValueQuality	MeasurementValueQuality	inherited from: MeasurementValue
[0..*]	[1..1] MeasurementValueSource	MeasurementValueSource	inherited from: MeasurementValue
[0..1]	[0..*] DiagramObjects	DiagramObject	inherited from: IdentifiedObject
[1..1]	[0..*] Names	Name	inherited from: IdentifiedObject

6.8.45 ReadingQuality root class

Quality of a specific reading value or interval reading value. Note that more than one quality may be applicable to a given reading. Typically not used unless problems or unusual conditions occur (i.e., quality for each reading is assumed to be good unless stated otherwise in associated reading quality type). It can also be used with the corresponding reading quality type to indicate that the validation has been performed and succeeded.

Table 195 shows all attributes of ReadingQuality.

Table 195 – Attributes of Metering::ReadingQuality

name	type	description
timeStamp	DateTime	Date and time at which the quality code was assigned or ascertained.
source	String	System acting as the source of the quality code.
comment	String	Elaboration on the quality code.

Table 196 shows all association ends of ReadingQuality with other classes.

Table 196 – Association ends of Metering::ReadingQuality with other classes

[mult from]	[mult to] name	type	description
[0..*]	[0..1] Reading	BaseReading	Reading value to which this quality applies.
[0..*]	[1..1] ReadingQualityType	ReadingQualityType	Type of this reading quality.

6.8.46 ReadingQualityType

Detailed description for a quality of a reading value, produced by an end device or a system. Values in attributes allow for creation of the recommended codes to be used for identifying reading value quality codes as follows: <systemId>.<category>.<subCategory>.

Table 197 shows all attributes of ReadingQualityType.

Table 197 – Attributes of Metering::ReadingQualityType

name	type	description
systemId	String	Identification of the system which has declared the issue with the data or provided commentary on the data.
category	String	High-level nature of the reading value quality.
subCategory	String	More specific nature of the reading value quality, as a further sub-categorisation of 'category'.
aliasName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject

Table 198 shows all association ends of ReadingQualityType with other classes.

Table 198 – Association ends of Metering::ReadingQualityType with other classes

[mult from]	[mult to] name	type	description
[1..1]	[0..*] ReadingQualities	ReadingQuality	All reading qualities of this type.
[0..1]	[0..*] DiagramObjects	DiagramObject	inherited from: IdentifiedObject
[1..1]	[0..*] Names	Name	inherited from: IdentifiedObject

6.8.47 ReadingType

Detailed description for a type of a reading value. Values in attributes allow for the creation of recommended codes to be used for identifying reading value types as follows: <macroPeriod>.<aggregate>.<measuringPeriod>.<accumulation>.<flowDirection>.<commodity>.<measurementKind>.<interharmonic.numerator>.<interharmonic.denominator>.<argument.numerator>.<argument.denominator>.<tou>.<cpp>.<consumptionTier>.<phases>.<multiplier>.<unit>.<currency>.

Table 199 shows all attributes of ReadingType.

Table 199 – Attributes of Metering::ReadingType

name	type	description
macroPeriod	String	Time period of interest that reflects how the reading is viewed or captured over a long period of time.
aggregate	String	Salient attribute of the reading data aggregated from individual endpoints. This is mainly used to define a mathematical operation carried out over 'macroPeriod', but may also be used to describe an attribute of the data when the 'macroPeriod' is not defined.
measuringPeriod	String	Time attribute inherent or fundamental to the reading value (as opposed to 'macroPeriod' that supplies an "adjective" to describe aspects of a time period with regard to the measurement). It refers to the way the value was originally measured and not to the frequency at which it is reported or presented. For example, an hourly interval of consumption data would have value 'hourly' as an attribute. However in the case of an hourly sampled voltage value, the meterReadings schema would carry the 'hourly' interval size information. It is common for meters to report demand in a form that is measured over the course of a portion of an hour, while enterprise applications however commonly assume the demand (in kW or kVAr) normalised to 1 h. The sytem that receives readings directly from the meter therefore shall perform this transformation before publishing readings for use by the other enterprise systems. The scalar used is chosen based on the block size (not any sub-interval size).
accumulation	String	Accumulation behaviour of a reading over time, usually 'measuringPeriod', to be used with individual endpoints (as opposed to 'macroPeriod' and 'aggregate' that are used to describe aggregations of data from individual endpoints).
flowDirection	String	Flow direction for a reading where the direction of flow of the commodity is important (for electricity measurements this includes current, energy, power, and demand).
commodity	String	Commodity being measured.
measurementKind	String	Identifies "what" is being measured, as refinement of 'commodity'. When combined with 'unit', it provides detail to the unit of measure. For example, 'energy' with a unit of measure of 'kWh' indicates to the user that active energy is being measured, while with 'kVAh' or 'kVArh', it indicates apparent energy and reactive energy, respectively. 'power' can be combined in a similar way with various power units of measure: Distortion power ('distortionVoltAmperes') with 'kVA' is different from 'power' with 'kVA'.

name	type	description
interharmonic	ReadingInterharmonic	Indication of a "harmonic" or "interharmonic" basis for the measurement. Value 0 in 'numerator' and 'denominator' means not applicable.
argument	RationalNumber	Argument used to introduce numbers into the unit of measure description where they are needed (e.g., 4 where the measure needs an argument such as CEMI(n=4)). Most arguments used in practice however will be integers (i.e., 'denominator'=1). Value 0 in 'numerator' and 'denominator' means not applicable.
tou	Integer	Time of use (TOU) bucket the reading value is attributed to. Value 0 means not applicable.
cpp	Integer	Critical peak period (CPP) bucket the reading value is attributed to. Value 0 means not applicable. Even though CPP is usually considered a specialised form of time of use 'tou', this attribute is defined explicitly for flexibility.
consumptionTier	Integer	In case of common flat-rate pricing for power, in which all purchases are at a given rate, 'consumptionTier'=0. Otherwise, the value indicates the consumption tier, which can be used in conjunction with TOU or CPP pricing. Consumption tier pricing refers to the method of billing in which a certain "block" of energy is purchased/sold at one price, after which the next block of energy is purchased at another price, and so on, all throughout a defined period. At the start of the defined period, consumption is initially zero, and any usage is measured against the first consumption tier ('consumptionTier'=1). If this block of energy is consumed before the end of the period, energy consumption moves to be reconed against the second consumption tier ('consumptionTier'=2), and so on. At the end of the defined period, the consumption accumulator is reset, and usage within the 'consumptionTier'=1 restarts.
phases	String	Meteriing-specific phase code.
multiplier	String	Metering-specific multiplier.
unit	String	Metering-specific unit.
currency	String	Metering-specific currency.
aliasName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject

Table 200 shows all association ends of ReadingType with other classes.

Table 200 – Association ends of Metering::ReadingType with other classes

[mult from]	[mult to] name	type	description
[0..1]	[0..1] Channel	Channel	Channel reporting/collecting register values with this type information.
[1..1]	[0..*] IntervalBlocks	IntervalBlock	All blocks containing interval reading values with this type information.
[1..*]	[0..*] MetrologyRequirements	MetrologyRequirement	All metrology requirements that require this reading type to be collected.
[1..1]	[0..1] PendingCalculation	PendingCalculation	Pending calculation that produced this reading type.
[1..1]	[0..*] Readings	Reading	All reading values with this type information.
[0..1]	[0..*] DiagramObjects	DiagramObject	inherited from: IdentifiedObject
[1..1]	[0..*] Names	Name	inherited from: IdentifiedObject

6.8.48 Register

A device that indicates or records units of the commodity or other quantity measured.

Table 201 shows all attributes of Register.

Table 201 – Attributes of Metering::Register

name	type	description
isVirtual	Boolean	If true, the data it produces is calculated or measured by a device other than a physical end device/meter. Otherwise, any data streams it produces are measured by the hardware of the end device/meter itself.
rightDigitCount	Integer	Number of digits (dials on a mechanical meter) to the right of the decimal place.
leftDigitCount	Integer	Number of digits (dials on a mechanical meter) to the left of the decimal place; default is normally 5.
touTier	TimeInterval	Clock time interval for register to beging/cease accumulating time of usage (e.g., start at 8:00 am, stop at 5:00 pm).
touTierName	String	Name used for the time of use tier (also known as bin or bucket). For example, "peak", "off-peak", "TOU Category A", etc.
aliasName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject

Table 202 shows all association ends of Register with other classes.

Table 202 – Association ends of Metering::Register with other classes

[mult from]	[mult to] name	type	description
[0..1]	[0..*] Channels	Channel	All channels that collect/report values from this register.
[0..*]	[0..1] EndDeviceFunction	EndDeviceFunction	End device function metering quantities displayed by this register.
[0..1]	[0..*] DiagramObjects	DiagramObject	inherited from: IdentifiedObject
[1..1]	[0..*] Names	Name	inherited from: IdentifiedObject

6.8.49 ServiceMultiplier

Multiplier applied at the usage point.

Table 203 shows all attributes of ServiceMultiplier.

Table 203 – Attributes of Metering::ServiceMultiplier

name	type	description
kind	ServiceMultiplierKind	Kind of multiplier.
value	Float	Multiplier value.
aliasName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject

Table 204 shows all association ends of ServiceMultiplier with other classes.

Table 204 – Association ends of Metering::ServiceMultiplier with other classes

[mult from]	[mult to] name	type	description
[0..*]	[0..1] UsagePoint	UsagePoint	Usage point applying this multiplier.
[0..1]	[0..*] DiagramObjects	DiagramObject	inherited from: IdentifiedObject
[1..1]	[0..*] Names	Name	inherited from: IdentifiedObject

6.8.50 SimpleEndDeviceFunction

Simple end device function distinguished by 'kind'. Use this class for instances that cannot be represented by another end device function specialisations.

Table 205 shows all attributes of SimpleEndDeviceFunction.

Table 205 – Attributes of Metering::SimpleEndDeviceFunction

name	type	description
kind	EndDeviceFunctionKind	Kind of this function.
enabled	Boolean	inherited from: EndDeviceFunction
programID	String	inherited from: AssetFunction
firmwareID	String	inherited from: AssetFunction
hardwareID	String	inherited from: AssetFunction
password	String	inherited from: AssetFunction
configID	String	inherited from: AssetFunction
aliasName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject

Table 206 shows all association ends of SimpleEndDeviceFunction with other classes.

Table 206 – Association ends of Metering::SimpleEndDeviceFunction with other classes

[mult from]	[mult to] name	type	description
[0..*]	[0..1] EndDevice	EndDevice	inherited from: EndDeviceFunction
[0..1]	[0..*] Registers	Register	inherited from: EndDeviceFunction
[0..1]	[0..*] DiagramObjects	DiagramObject	inherited from: IdentifiedObject
[1..1]	[0..*] Names	Name	inherited from: IdentifiedObject

6.8.51 UsagePoint

Logical or physical point in the network to which readings or events may be attributed. Used at the place where a physical or virtual meter may be located; however, it is not required that a meter be present.

Table 207 shows all attributes of UsagePoint.

Table 207 – Attributes of Metering::UsagePoint

name	type	description
isSdp	Boolean	If true, this usage point is a service delivery point, i.e., a usage point where the ownership of the service changes hands.
isVirtual	Boolean	If true, this usage point is virtual, i.e., no physical location exists in the network where a meter could be located to collect the meter readings. For example, one may define a virtual usage point to serve as an aggregation of usage for all of a company's premises distributed widely across the distribution territory. Otherwise, the usage point is physical, i.e., there is a logical point in the network where a meter could be located to collect meter readings.
phaseCode	PhaseCode	Phase code. Number of wires and specific nominal phases can be deduced from enumeration literal values. For example, ABCN is three-phase, four-wire, s12n (splitSecondary12N) is single-phase, three-wire, and s1n and s2n are single-phase, two-wire.
grounded	Boolean	True if grounded.
servicePriority	String	Priority of service for this usage point. Note that usage points at the same service location can have different priorities.
serviceDeliveryRemark	String	Remarks about this usage point, for example the reason for it being rated with a non-nominal priority.
estimatedLoad	CurrentFlow	Estimated load.
checkBilling	Boolean	True if as a result of an inspection or otherwise, there is a reason to suspect that a previous billing may have been performed with erroneous data. Value should be reset once this potential discrepancy has been resolved.
ratedCurrent	CurrentFlow	Current flow that this usage point is configured to deliver.
nominalServiceVoltage	Voltage	Nominal service voltage.
ratedPower	ActivePower	Active power that this usage point is configured to deliver.
outageRegion	String	Outage region in which this usage point is located.
readCycle	String	Cycle day on which the meter for this usage point will normally be read. Usually correlated with the billing cycle.
readRoute	String	Identifier of the route to which this usage point is assigned for purposes of meter reading. Typically used to configure hand held meter reading systems prior to collection of reads.
amiBillingReady	AmiBillingReadyKind	Tracks the lifecycle of the metering installation at a usage point with respect to readiness for billing via advanced metering infrastructure reads.
connectionState	UsagePointConnectedKind	State of the usage point with respect to connection to the network.
minimalUsageExpected	Boolean	If true, minimal or zero usage is expected at this usage point for situations such as premises vacancy, logical or physical disconnect. It is used for readings validation and estimation.
aliasName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject

Table 208 shows all association ends of UsagePoint with other classes.

Table 208 – Association ends of Metering::UsagePoint with other classes

[mult from]	[mult to] name	type	description
[0..*]	[0..*] Equipments	Equipment	All equipment connecting this usage point to the electrical grid.
[0..1]	[0..*] ConfigurationEvents	ConfigurationEvent	All configuration events created for this usage point.
[0..*]	[0..1] ServiceCategory	ServiceCategory	Service category delivered by this usage point.
[0..1]	[0..*] EndDevices	EndDevice	All end devices at this usage point.
[0..1]	[0..*] ServiceMultipliers	ServiceMultiplier	All multipliers applied at this usage point.
[0..*]	[0..1] CustomerAgreement	CustomerAgreement	Customer agreement regulating this service delivery point.
[0..*]	[0..*] PricingStructures	PricingStructure	All pricing structures applicable to this service delivery point (with prepayment meter running as a stand-alone device, with no CustomerAgreement or Customer).
[0..*]	[0..1] ServiceLocation	ServiceLocation	Service location where the service delivered by this usage point is consumed.
[0..*]	[0..*] EndDeviceControls	EndDeviceControl	All end device controls sending commands to this usage point.
[0..1]	[0..*] EndDeviceEvents	EndDeviceEvent	All end device events reported for this usage point.
[0..1]	[0..*] MeterReadings	MeterReading	All meter readings obtained from this usage point.
[0..1]	[0..*] MeterServiceWorks	MeterServiceWork	All meter service works at this usage point.
[0..*]	[0..*] MetrologyRequirements	MetrologyRequirement	All metrology requirements for this usage point.
[0..*]	[0..1] ServiceSupplier	ServiceSupplier	ServiceSupplier (utility) utilising this usage point to deliver a service.
[0..*]	[0..*] UsagePointGroups	UsagePointGroup	All groups to which this usage point belongs.
[0..*]	[0..1] UsagePointLocation	UsagePointLocation	Location of this usage point.
[0..1]	[0..*] DiagramObjects	DiagramObject	inherited from: IdentifiedObject
[1..1]	[0..*] Names	Name	inherited from: IdentifiedObject

6.8.52 UsagePointGroup

Abstraction for management of group communications within a two-way AMR system or the data for a group of related usage points. Commands can be issued to all of the usage points that belong to a usage point group using a defined group address and the underlying AMR communication infrastructure.

Table 209 shows all attributes of UsagePointGroup.

Table 209 – Attributes of Metering::UsagePointGroup

name	type	description
aliasName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject

Table 210 shows all association ends of UsagePointGroup with other classes.

Table 210 – Association ends of Metering::UsagePointGroup with other classes

[mult from]	[mult to] name	type	description
[0..*]	[0..*] UsagePoints	UsagePoint	All usage points in this group.
[0..*]	[0..*] DemandResponsePrograms	DemandResponseProgram	All demand response programs this usage point group is enrolled in.
[0..*]	[0..*] EndDeviceControls	EndDeviceControl	All end device controls sending commands to this usage point group.
[0..1]	[0..*] DiagramObjects	DiagramObject	inherited from: IdentifiedObject
[1..1]	[0..*] Names	Name	inherited from: IdentifiedObject

6.8.53 UsagePointLocation

Location of an individual usage point.

Table 211 shows all attributes of UsagePointLocation.

Table 211 – Attributes of Metering::UsagePointLocation

name	type	description
accessMethod	String	Method for the service person to access this usage point location. For example, a description of where to obtain a key if the facility is unmanned and secured.
siteAccessProblem	String	Problems previously encountered when visiting or performing work at this location. Examples include: bad dog, violent customer, verbally abusive occupant, obstructions, safety hazards, etc.
remark	String	Remarks about this location.
type	String	inherited from: Location
mainAddress	StreetAddress	inherited from: Location
secondaryAddress	StreetAddress	inherited from: Location
phone1	TelephoneNumber	inherited from: Location
phone2	TelephoneNumber	inherited from: Location
electronicAddress	ElectronicAddress	inherited from: Location
geoInfoReference	String	inherited from: Location
direction	String	inherited from: Location
status	Status	inherited from: Location
aliasName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject

Table 212 shows all association ends of UsagePointLocation with other classes.

Table 212 – Association ends of Metering::UsagePointLocation with other classes

[mult from]	[mult to] name	type	description
[0..1]	[0..*] UsagePoints	UsagePoint	All usage points at this location.
[0..1]	[0..*] PowerSystemResources	PowerSystemResource	inherited from: Location
[0..1]	[0..*] ConfigurationEvents	ConfigurationEvent	inherited from: Location
[0..*]	[0..1] CoordinateSystem	CoordinateSystem	inherited from: Location
[1..1]	[0..*] PositionPoints	PositionPoint	inherited from: Location
[0..1]	[0..*] Assets	Asset	inherited from: Location
[0..1]	[0..*] DiagramObjects	DiagramObject	inherited from: IdentifiedObject
[1..1]	[0..*] Names	Name	inherited from: IdentifiedObject

6.9 Package LoadControl

6.9.1 General

This package is an extension of the Metering package and contains the information classes that support specialised applications such as demand-side management using load control equipment. These classes are generally associated with the point where a service is delivered to the customer.

Figure 52 shows class diagram LoadControlInheritance.

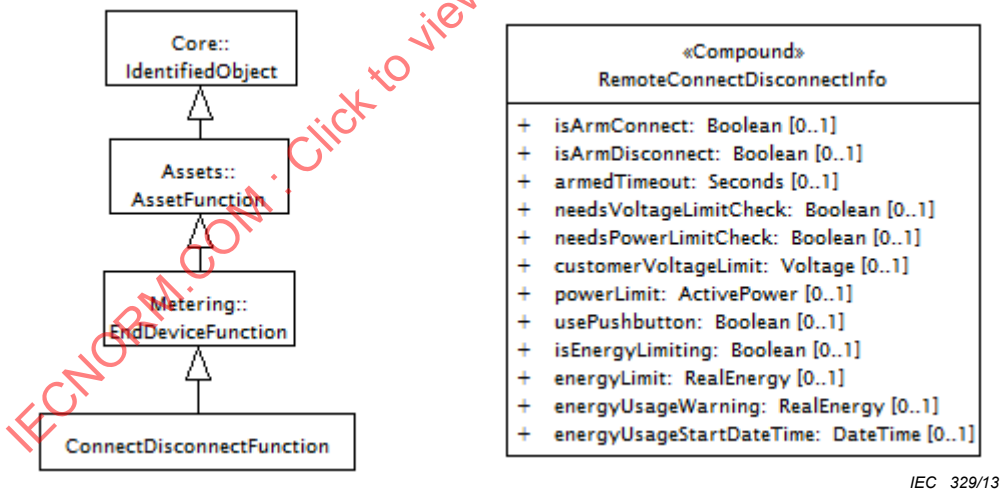


Figure 52 – Class diagram LoadControl::LoadControlInheritance

This diagram shows the inheritance hierarchy for normative classes from this package, as well as enumerations and compound types.

Figure 53 shows class diagram LoadControlOverview.

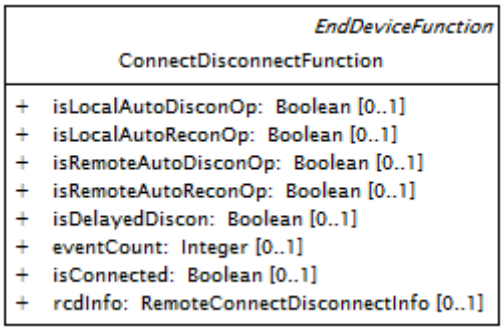


Figure 53 – Class diagram LoadControl::LoadControlOverview

This diagram shows normative classes from this package.

6.9.2 RemoteConnectDisconnectInfo compound

Details of remote connect and disconnect function.

Table 213 shows all attributes of RemoteConnectDisconnectInfo.

Table 213 – Attributes of LoadControl::RemoteConnectDisconnectInfo

name	type	description
isArmConnect	Boolean	True if the RCD switch has to be armed before a connect action can be initiated.
isArmDisconnect	Boolean	True if the RCD switch has to be armed before a disconnect action can be initiated.
armedTimeout	Seconds	Setting of the timeout elapsed time.
needsVoltageLimitCheck	Boolean	True if voltage limit has to be checked to prevent connect if voltage is over the limit.
needsPowerLimitCheck	Boolean	True if load limit has to be checked to issue an immediate disconnect (after a connect) if load is over the limit.
customerVoltageLimit	Voltage	Voltage limit on customer side of RCD switch above which the connect should not be made.
powerLimit	ActivePower	Load limit above which the connect should either not take place or should cause an immediate disconnect.
usePushbutton	Boolean	True if pushbutton has to be used for connect.
isEnergyLimiting	Boolean	True if the energy usage is limited and the customer will be disconnected if they go over the limit.
energyLimit	RealEnergy	Limit of energy before disconnect.
energyUsageWarning	RealEnergy	Warning energy limit, used to trigger event code that energy usage is nearing limit.
energyUsageStartDateTime	DateTime	Start date and time to accumulate energy for energy usage limiting.

6.9.3 ConnectDisconnectFunction

A function that will disconnect and reconnect the customer's load under defined conditions.

Table 214 shows all attributes of ConnectDisconnectFunction.

Table 214 – Attributes of LoadControl::ConnectDisconnectFunction

name	type	description
isLocalAutoDisconOp	Boolean	If set true and if disconnection can be operated locally, the operation happens automatically. Otherwise it happens manually.
isLocalAutoReconOp	Boolean	If set true and if reconnection can be operated locally, then the operation happens automatically. Otherwise, it happens manually.
isRemoteAutoDisconOp	Boolean	If set true and if disconnection can be operated remotely, then the operation happens automatically. If set false and if disconnection can be operated remotely, then the operation happens manually.
isRemoteAutoReconOp	Boolean	If set true and if reconnection can be operated remotely, then the operation happens automatically. If set false and if reconnection can be operated remotely, then the operation happens manually.
isDelayedDiscon	Boolean	If set true, the switch may disconnect the service at the end of a specified time delay after the disconnect signal has been given. If set false, the switch may disconnect the service immediately after the disconnect signal has been given. This is typically the case for over current circuit-breakers which are classified as either instantaneous or slow acting.
eventCount	Integer	Running cumulative count of connect or disconnect events, for the lifetime of this function or until the value is cleared.
isConnected	Boolean	True if this function is in the connected state.
rcdInfo	RemoteConnectDisconnectInfo	Information on remote connect disconnect switch.
enabled	Boolean	inherited from: EndDeviceFunction
programID	String	inherited from: AssetFunction
firmwareID	String	inherited from: AssetFunction
hardwareID	String	inherited from: AssetFunction
password	String	inherited from: AssetFunction
configID	String	inherited from: AssetFunction
aliasName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject

Table 215 shows all association ends of ConnectDisconnectFunction with other classes.

Table 215 – Association ends of LoadControl::ConnectDisconnectFunction with other classes

[mult from]	[mult to] name	type	description
[0..*]	[0..1] EndDevice	EndDevice	inherited from: EndDeviceFunction
[0..1]	[0..*] Registers	Register	inherited from: EndDeviceFunction
[0..1]	[0..*] DiagramObjects	DiagramObject	inherited from: IdentifiedObject
[1..1]	[0..*] Names	Name	inherited from: IdentifiedObject

6.10 Package PaymentMetering

6.10.1 General

This package is an extension of the Metering package and contains the information classes that support specialised applications such as prepayment metering. These classes are generally associated with the collection and control of revenue from the customer for a delivered service.

Figure 54 shows class diagram PaymentMeteringInheritance.

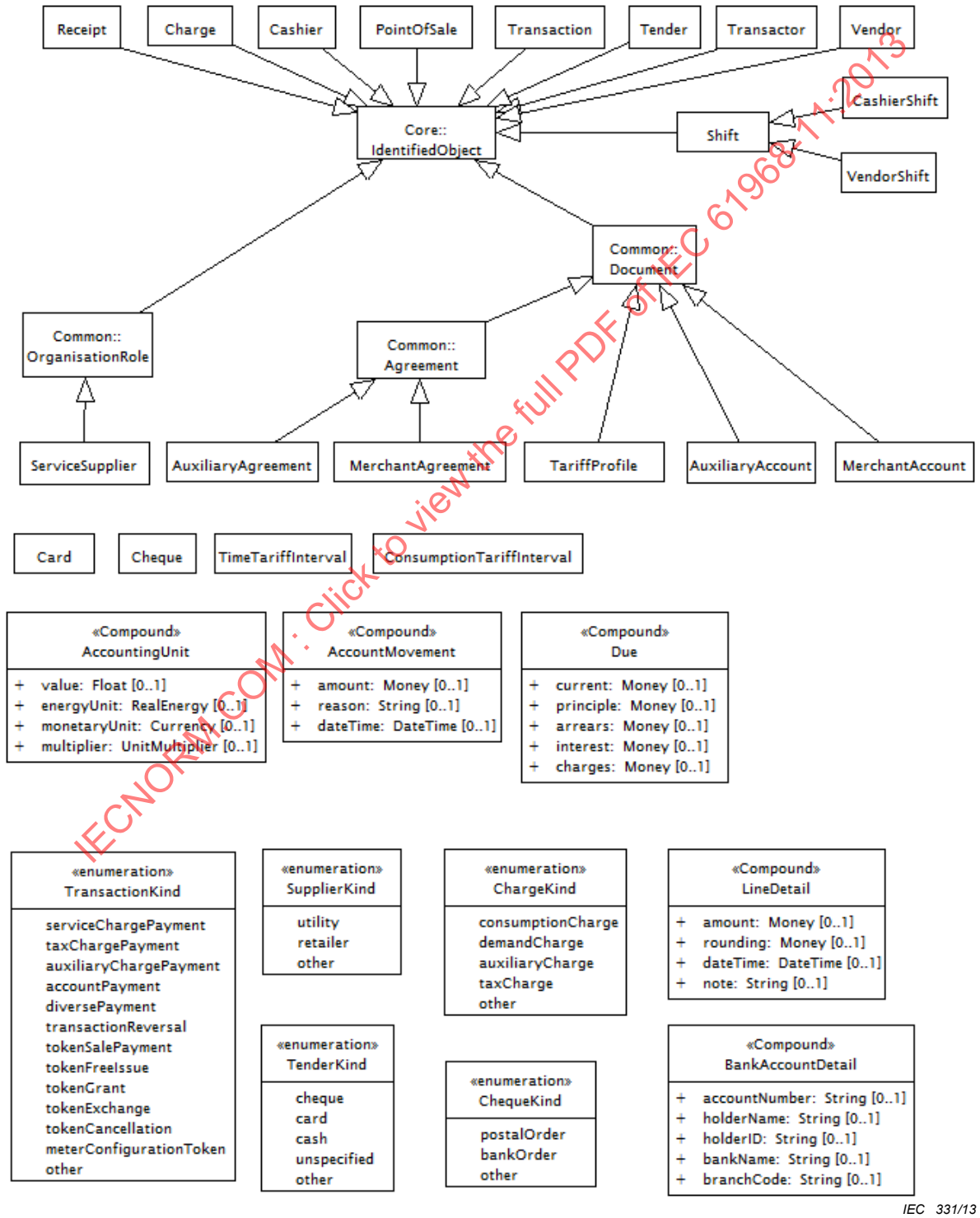


Figure 54 – Class diagram PaymentMetering::PaymentMeteringInheritance

This diagram shows the inheritance hierarchy for normative classes from this package, as well as enumerations and compound types.

Figure 55 shows class diagram PaymentMeteringOverview.

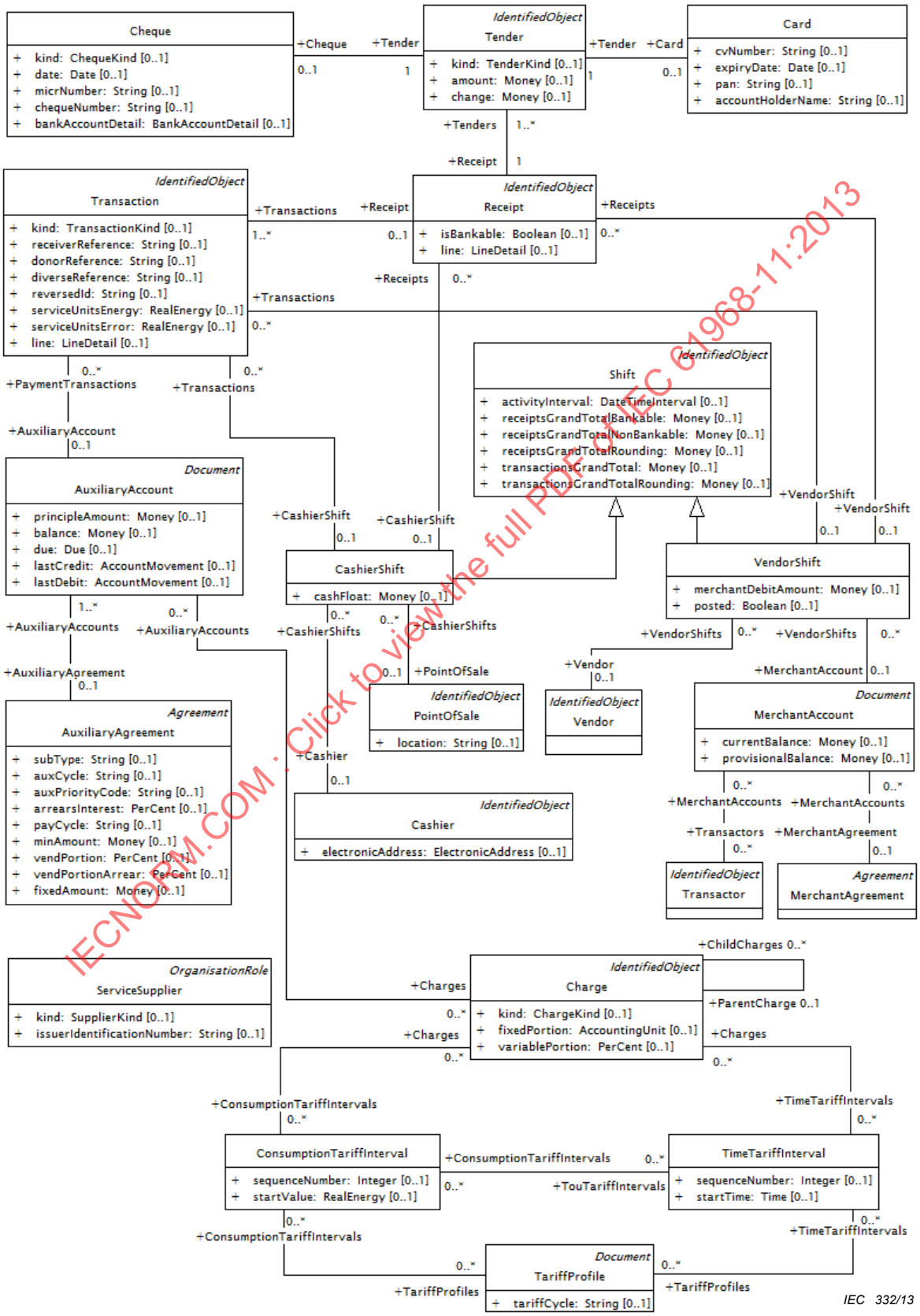


Figure 55 – Class diagram PaymentMetering::PaymentMeteringOverview

This diagram shows normative classes from this package.

Figure 56 shows class diagram PaymentMeteringRelationships.

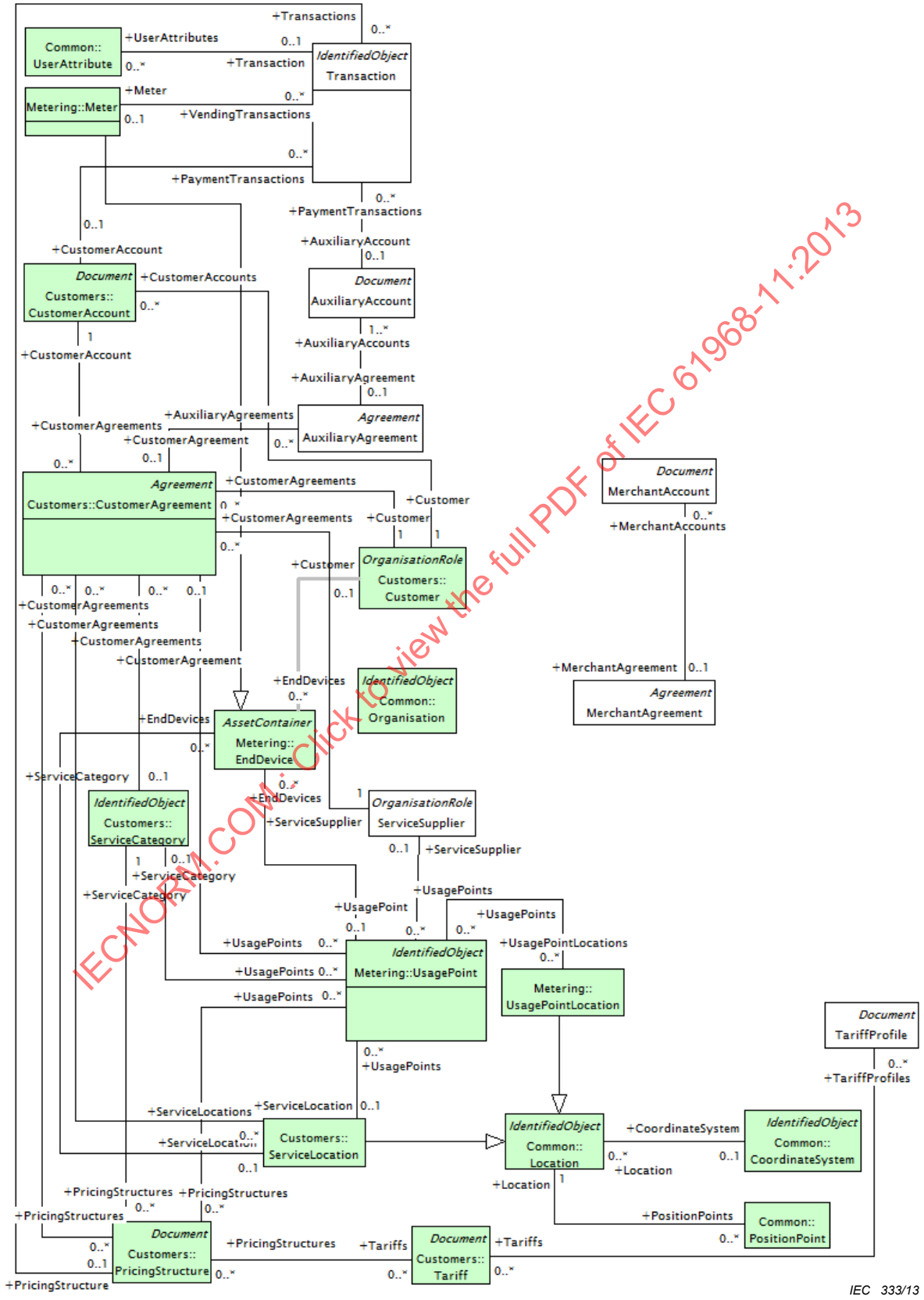


Figure 56 – Class diagram PaymentMetering::PaymentMeteringRelationships

This diagram shows relationships of the classes from this package with those from other packages.

Figure 57 shows class diagram Transacting.

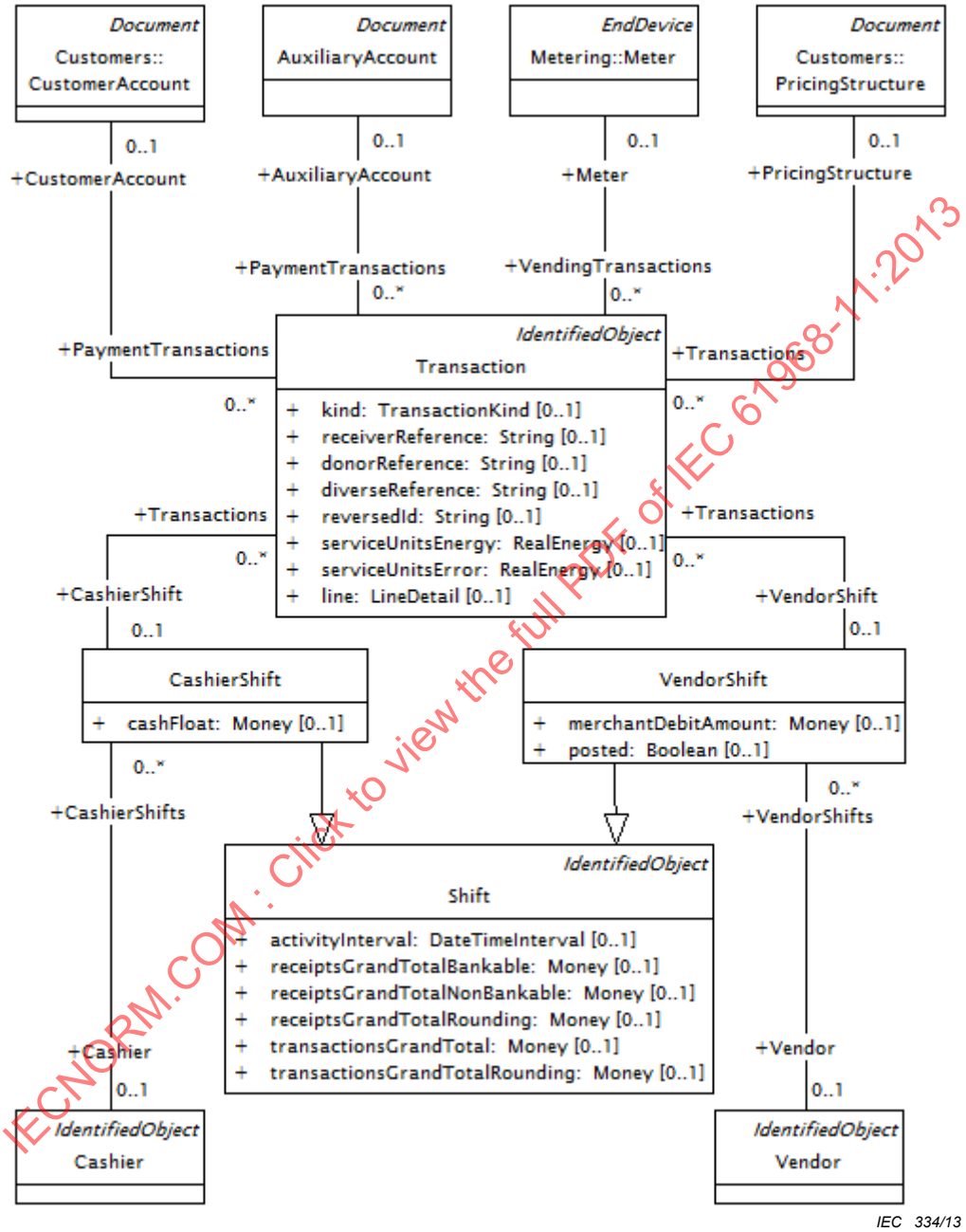


Figure 57 – Class diagram PaymentMetering::Transacting

This diagram shows classes used to model transacting.

Figure 58 shows class diagram Receipting.

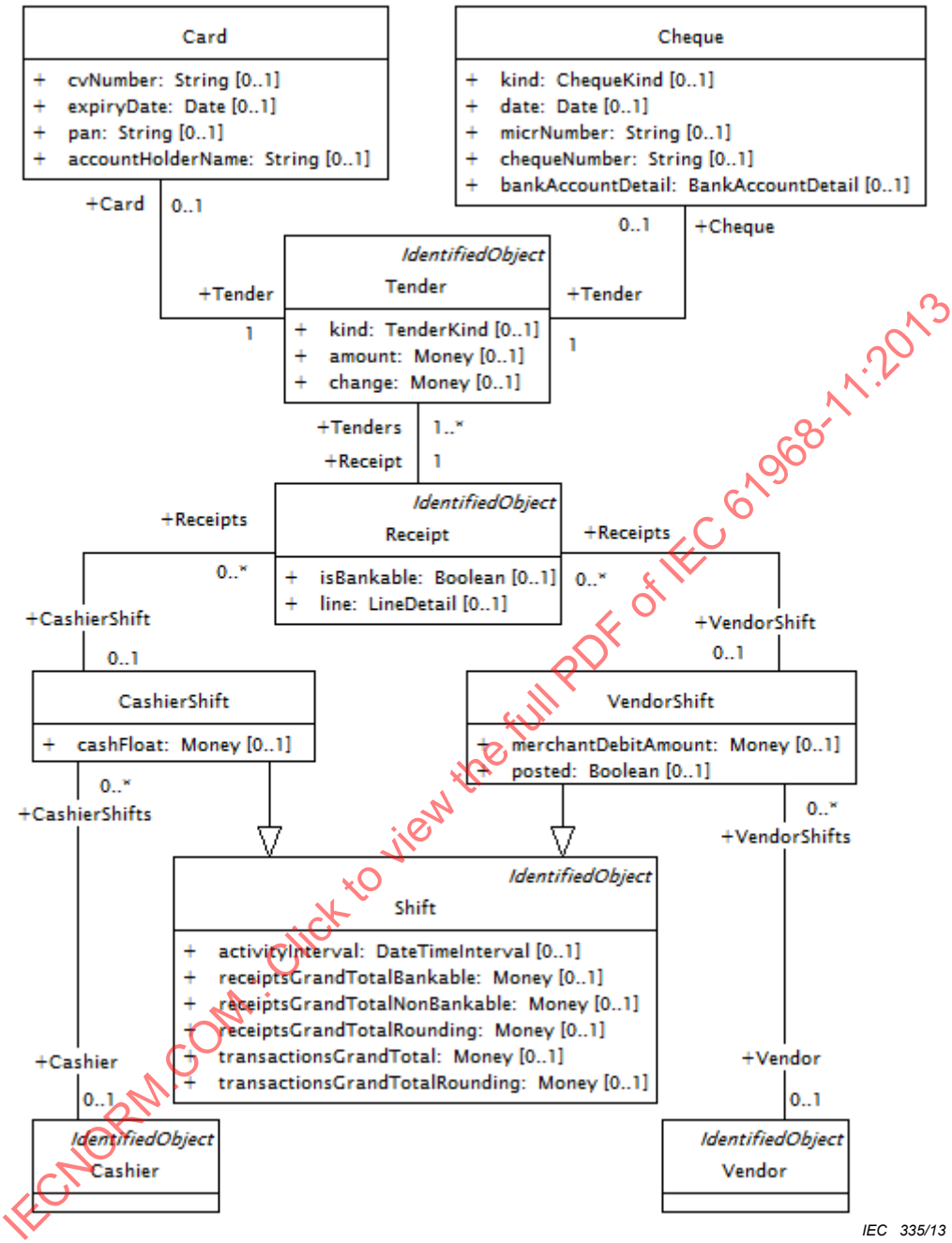


Figure 58 – Class diagram PaymentMetering::Receipting

This diagram shows classes used to model receipting.

Figure 59 shows class diagram AuxiliaryAgreement.

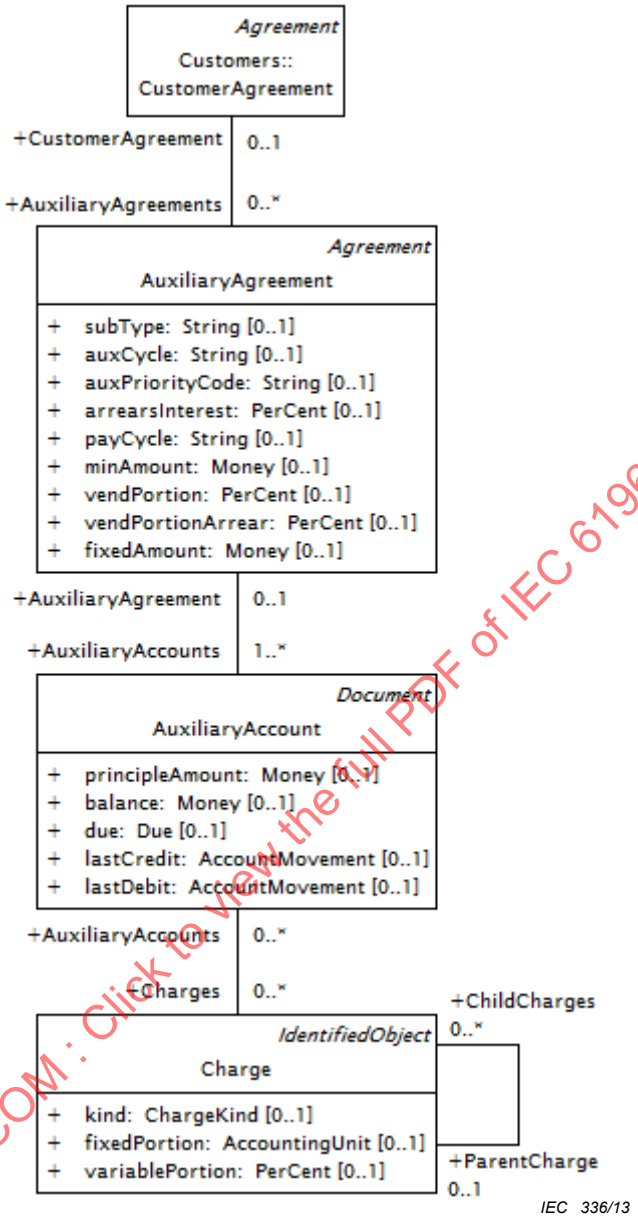


Figure 59 – Class diagram PaymentMetering::AuxiliaryAgreement

This diagram shows classes used to model auxiliary agreements.

Figure 60 shows class diagram TariffProfile.

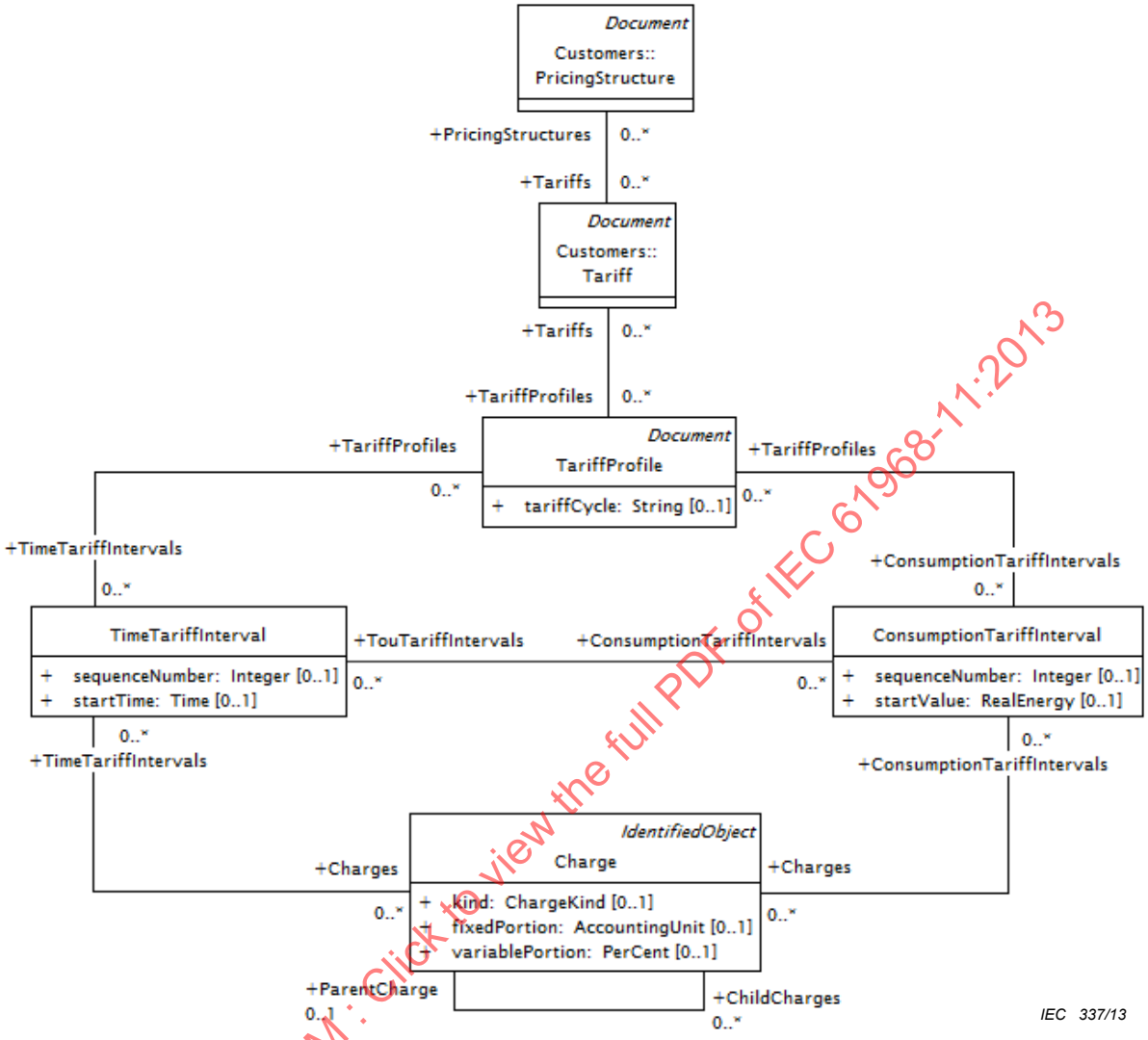


Figure 60 – Class diagram PaymentMetering::TariffProfile

This diagram shows classes used to model tariff profiles.

6.10.2 AccountMovement compound

Credit/debit movements for an account.

Table 216 shows all attributes of AccountMovement.

Table 216 – Attributes of PaymentMetering::AccountMovement

name	type	description
amount	Money	Amount that was credited to/debited from an account. For example: payment received/interest charge on arrears.
reason	String	Reason for credit/debit transaction on an account. Example: payment received/arrears interest levied.
dateTime	DateTime	Date and time when the credit/debit transaction was performed.

6.10.3 AccountingUnit compound

Unit for accounting; use either 'energyUnit' or 'currencyUnit' to specify the unit for 'value'.

Table 217 shows all attributes of AccountingUnit.

Table 217 – Attributes of PaymentMetering::AccountingUnit

name	type	description
value	Float	Value expressed in applicable units.
energyUnit	RealEnergy	Unit of service.
monetaryUnit	Currency	Unit of currency.
multiplier	UnitMultiplier	Multiplier for the 'energyUnit' or 'monetaryUnit'.

6.10.4 BankAccountDetail compound

Details of a bank account.

Table 218 shows all attributes of BankAccountDetail.

Table 218 – Attributes of PaymentMetering::BankAccountDetail

name	type	description
accountNumber	String	Operational account reference number.
holderName	String	Name of account holder.
holderID	String	National identity number (or equivalent) of account holder.
bankName	String	Name of bank where account is held.
branchCode	String	Branch of bank where account is held.

6.10.5 Due compound

Details on amounts due for an account.

Table 219 shows all attributes of Due.

Table 219 – Attributes of PaymentMetering::Due

name	type	description
current	Money	Current total amount now due: current = principle + arrears + interest + charges. Typically the rule for settlement priority is: interest dues, then arrears dues, then current dues, then charge dues.
principle	Money	Part of 'current' that constitutes the portion of the principle amount currently due.
arrears	Money	Part of 'current' that constitutes the arrears portion.
interest	Money	Part of 'current' that constitutes the interest portion.
charges	Money	Part of 'current' that constitutes the charge portion: 'charges' = 'Charge.fixedPortion' + 'Charge.variablePortion'.

6.10.6 LineDetail compound

Details on an amount line, with rounding, date and note.

Table 220 shows all attributes of LineDetail.

Table 220 – Attributes of PaymentMetering::LineDetail

name	type	description
amount	Money	Amount for this line item.
rounding	Money	Totalised monetary value of all errors due to process rounding or truncating that is not reflected in 'amount'.
dateTime	DateTime	Date and time when this line was created in the application process.
note	String	Free format note relevant to this line.

6.10.7 ChargeKind enumeration

Kind of charge.

Table 221 shows all literals of ChargeKind.

Table 221 – Literals of PaymentMetering::ChargeKind

literal	description
consumptionCharge	The charge levied for the actual usage of the service, normally expressed in terms of a tariff. For example: usage x price per kWh = total charge for consumption.
demandCharge	The charge related to the usage within a defined time interval, normally expressed in terms of a tariff. For example: a maximum-demand tariff will levy an additional charge on top of the consumption charge if the usage exceeds a defined limit per hour.
auxiliaryCharge	Any other charge which is not a consumptionCharge or demandCharge. For example: debt recovery, arrears, standing charge or charge for another service such as street lighting.
taxCharge	Any charge that is classified as a tax of a kind. For example: VAT, GST, TV tax, etc.
other	Other kind of charge.

6.10.8 ChequeKind enumeration

Kind of cheque.

Table 222 shows all literals of ChequeKind.

Table 222 – Literals of PaymentMetering::ChequeKind

literal	description
postalOrder	Payment order used by institutions other than banks.
bankOrder	Payment order used by a bank.
other	Other kind of cheque.

6.10.9 SupplierKind enumeration

Kind of supplier.

Table 223 shows all literals of SupplierKind.

Table 223 – Literals of PaymentMetering::SupplierKind

literal	description
utility	Entity that delivers the service to the customer.
retailer	Entity that sells the service, but does not deliver to the customer; applies to the deregulated markets.
other	Other kind of supplier.

6.10.10 TenderKind enumeration

Kind of tender.

Table 224 shows all literals of TenderKind.

Table 224 – Literals of PaymentMetering::TenderKind

literal	description
cheque	Payment method by means of a cheque.
card	Payment method by means of a credit or debit card.
cash	Payment method by means of cash.
unspecified	Payment method is not known.
other	Other payment method such as electronic finds transfer.

6.10.11 TransactionKind enumeration

Kind of transaction.

Table 225 shows all literals of TransactionKind.

Table 225 – Literals of PaymentMetering::TransactionKind

literal	description
serviceChargePayment	Payment for a service.
taxChargePayment	Payment for a tax.
auxiliaryChargePayment	Payment against a specified auxiliary account.
accountPayment	Payment against a specified account.
diversePayment	Payment against an item other than an account.
transactionReversal	Reversal of a previous transaction.
tokenSalePayment	Payment for a credit token sale to a customer.
tokenFreeIssue	Issue of a free credit token where the donor is the supplier.
tokenGrant	Issue of a free credit token where the donor is a 3rd party.
tokenExchange	Exchange of a previously issued token for a new token.
tokenCancellation	Cancellation of a previously issued token.
meterConfigurationToken	Issue of token that will alter the meter configuration.
other	Other kind of transaction.

6.10.12 AuxiliaryAccount

Variable and dynamic part of auxiliary agreement, generally representing the current state of the account related to the outstanding balance defined in auxiliary agreement.

Table 226 shows all attributes of AuxiliaryAccount.

Table 226 – Attributes of PaymentMetering::AuxiliaryAccount

name	type	description
principleAmount	Money	The initial principle amount, with which this account was instantiated.
balance	Money	The total amount currently remaining on this account that is required to be paid in order to settle the account to zero. This excludes any due amounts not yet paid.
due	Due	Current amounts now due for payment on this account.
lastCredit	AccountMovement	Details of the last credit transaction performed on this account.
lastDebit	AccountMovement	Details of the last debit transaction performed on this account.
type	String	inherited from: Document
createdDateTime	DateTime	inherited from: Document
lastModifiedDateTime	DateTime	inherited from: Document
revisionNumber	String	inherited from: Document
electronicAddress	ElectronicAddress	inherited from: Document
subject	String	inherited from: Document
title	String	inherited from: Document
docStatus	Status	inherited from: Document
status	Status	inherited from: Document
aliasName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject

Table 227 shows all association ends of AuxiliaryAccount with other classes.

Table 227 – Association ends of PaymentMetering::AuxiliaryAccount with other classes

[mult from]	[mult to] name	type	description
[0..1]	[0..*] PaymentTransactions	Transaction	All payments against this account.
[0..*]	[0..*] Charges	Charge	All charges levied on this account.
[1..*]	[0..1] AuxiliaryAgreement	AuxiliaryAgreement	Auxiliary agreement regulating this account.
[0..1]	[0..*] ConfigurationEvents	ConfigurationEvent	inherited from: Document
[0..1]	[0..*] DiagramObjects	DiagramObject	inherited from: IdentifiedObject
[1..1]	[0..*] Names	Name	inherited from: IdentifiedObject

6.10.13 AuxiliaryAgreement

An ad-hoc auxiliary account agreement associated with a customer agreement, not part of the customer's account, but typically subject to formal agreement between customer and supplier (utility). Typically this is used to collect revenue owed by the customer for other services or

arrears accrued with the utility for other services. It is typically linked to a prepaid token purchase transaction, thus forcing the customer to make a payment towards settlement of the auxiliary account balance whenever the customer needs to purchase a prepaid token for electricity.

The present status of the auxiliary agreement can be defined in the context of the utility's business rules, for example: enabled, disabled, pending, over recovered, under recovered, written off, etc.

Table 228 shows all attributes of AuxiliaryAgreement.

Table 228 – Attributes of PaymentMetering::AuxiliaryAgreement

name	type	description
subType	String	Sub-classification of the inherited 'type' for this AuxiliaryAgreement.
auxCycle	String	The frequency for automatically recurring auxiliary charges, where 'AuxiliaryAccount.initialCharge' is recursively added to 'AuxiliaryAccount.dueCurrent' at the start of each 'auxCycle'. For example: on a specified date and time; hourly; daily; weekly; monthly; 3-monthly; 6-monthly; 12-monthly; etc.
auxPriorityCode	String	The coded priority indicating the priority that this auxiliary agreement has above other auxiliary agreements (associated with the same customer agreement) when it comes to competing for settlement from a payment transaction or token purchase.
arrearsInterest	PerCent	The interest per annum to be charged prorata on 'AuxiliaryAccount.dueArrears' at the end of each 'payCycle'.
payCycle	String	The contractually expected payment frequency (by the customer). Examples are: ad-hoc; on specified date; hourly, daily, weekly, monthly, etc.
minAmount	Money	The minimum amount that has to be paid at any transaction towards settling this auxiliary agreement or reducing the balance.
vendPortion	PerCent	The percentage of the transaction amount that has to be collected from each vending transaction towards settlement of this auxiliary agreement when payments are not in arrears. Note that there may be multiple tokens vended per vending transaction, but this is not relevant.
vendPortionArrear	PerCent	The percentage of the transaction amount that has to be collected from each vending transaction towards settlement of this auxiliary agreement when payments are in arrears. Note that there may be multiple tokens vended per vending transaction, but this is not relevant.
fixedAmount	Money	The fixed amount that has to be collected from each vending transaction towards settlement of this auxiliary agreement. Note that there may be multiple tokens vended per vending transaction, but this is not relevant.
signDate	Date	inherited from: Agreement
validityInterval	DateTimeInterval	inherited from: Agreement
type	String	inherited from: Document
createdDateTime	DateTime	inherited from: Document
lastModifiedDateTime	DateTime	inherited from: Document
revisionNumber	String	inherited from: Document
electronicAddress	ElectronicAddress	inherited from: Document
subject	String	inherited from: Document
title	String	inherited from: Document
docStatus	Status	inherited from: Document
status	Status	inherited from: Document

name	type	description
aliasName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject

Table 229 shows all association ends of AuxiliaryAgreement with other classes.

Table 229 – Association ends of PaymentMetering::AuxiliaryAgreement with other classes

[mult from]	[mult to] name	type	description
[0..1]	[1..*] AuxiliaryAccounts	AuxiliaryAccount	All auxiliary accounts regulated by this agreement.
[0..*]	[0..1] CustomerAgreement	CustomerAgreement	Customer agreement this (non-service related) auxiliary agreement refers to.
[0..1]	[0..*] ConfigurationEvents	ConfigurationEvent	inherited from: Document
[0..1]	[0..*] DiagramObjects	DiagramObject	inherited from: IdentifiedObject
[1..1]	[0..*] Names	Name	inherited from: IdentifiedObject

6.10.14 Card root class

Documentation of the tender when it is a type of card (credit, debit, etc).

Table 230 shows all attributes of Card.

Table 230 – Attributes of PaymentMetering::Card

name	type	description
cvNumber	String	The card verification number.
expiryDate	Date	The date when this card expires.
pan	String	The primary account number.
accountHolderName	String	Name of account holder.

Table 231 shows all association ends of Card with other classes.

Table 231 – Association ends of PaymentMetering::Card with other classes

[mult from]	[mult to] name	type	description
[0..1]	[1..1] Tender	Tender	Payment tender this card is being used for.

6.10.15 Cashier

The operator of the point of sale for the duration of CashierShift. Cashier is under the exclusive management control of Vendor.

Table 232 shows all attributes of Cashier.

Table 232 – Attributes of PaymentMetering::Cashier

name	type	description
electronicAddress	ElectronicAddress	Electronic address.
aliasName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject

Table 233 shows all association ends of Cashier with other classes.

Table 233 – Association ends of PaymentMetering::Cashier with other classes

[mult from]	[mult to] name	type	description
[0..1]	[0..*] CashierShifts	CashierShift	All shifts operated by this cashier.
[0..1]	[0..*] DiagramObjects	DiagramObject	inherited from: IdentifiedObject
[1..1]	[0..*] Names	Name	inherited from: IdentifiedObject

6.10.16 CashierShift

The operating shift for a cashier, during which the cashier may transact against the cashier shift, subject to vendor shift being open.

Table 234 shows all attributes of CashierShift.

Table 234 – Attributes of PaymentMetering::CashierShift

name	type	description
cashFloat	Money	The amount of cash that the cashier brings to start the shift and that will be taken away at the end of the shift; i.e. the cash float does not get banked.
activityInterval	DateTimeInterval	inherited from: Shift
receiptsGrandTotalBankable	Money	inherited from: Shift
receiptsGrandTotalNonBankable	Money	inherited from: Shift
receiptsGrandTotalRounding	Money	inherited from: Shift
transactionsGrandTotal	Money	inherited from: Shift
transactionsGrandTotalRounding	Money	inherited from: Shift
aliasName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject

Table 235 shows all association ends of CashierShift with other classes.

Table 235 – Association ends of PaymentMetering::CashierShift with other classes

[mult from]	[mult to] name	type	description
[0..1]	[0..*] Receipts	Receipt	All Receipts recorded for this Shift.
[0..*]	[0..1] PointOfSale	PointOfSale	Point of sale that is in operation during this shift.
[0..1]	[0..*] Transactions	Transaction	All transactions recorded during this cashier shift.
[0..*]	[0..1] Cashier	Cashier	Cashier operating this shift.
[0..1]	[0..*] DiagramObjects	DiagramObject	inherited from: IdentifiedObject
[1..1]	[0..*] Names	Name	inherited from: IdentifiedObject

6.10.17 Charge

A charge element associated with other entities such as tariff structures, auxiliary agreements or other charge elements. The total charge amount applicable to this instance of charge is the sum of fixed and variable portion.

Table 236 shows all attributes of Charge.

Table 236 – Attributes of PaymentMetering::Charge

name	type	description
kind	ChargeKind	The kind of charge to be applied.
fixedPortion	AccountingUnit	The fixed portion of this charge element.
variablePortion	PerCent	The variable portion of this charge element, calculated as a percentage of the total amount of a parent charge.
aliasName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject

Table 237 shows all association ends of Charge with other classes.

Table 237 – Association ends of PaymentMetering::Charge with other classes

[mult from]	[mult to] name	type	description
[0..1]	[0..*] ChildCharges	Charge	All sub-components of this complex charge.
[0..*]	[0..*] AuxiliaryAccounts	AuxiliaryAccount	All auxiliary accounts to which this charge has to be levied.
[0..*]	[0..*] ConsumptionTariffIntervals	ConsumptionTariffInterval	Tariff intervals to which this consumption-based charge has to be levied.
[0..*]	[0..*] TimeTariffIntervals	TimeTariffInterval	Tariff intervals to which this time-based charge has to be levied.
[0..*]	[0..1] ParentCharge	Charge	Parent of this charge sub-component.
[0..1]	[0..*] DiagramObjects	DiagramObject	inherited from: IdentifiedObject
[1..1]	[0..*] Names	Name	inherited from: IdentifiedObject

6.10.18 Cheque root class

The actual tender when it is a type of cheque.

Table 238 shows all attributes of Cheque.

Table 238 – Attributes of PaymentMetering::Cheque

name	type	description
kind	ChequeKind	Kind of cheque.
date	Date	Date when cheque becomes valid.
micrNumber	String	The magnetic ink character recognition number printed on the cheque.
chequeNumber	String	Cheque reference number as printed on the cheque.
bankAccountDetail	BankAccountDetail	Details of the account holder and bank.

Table 239 shows all association ends of Cheque with other classes.

Table 239 – Association ends of PaymentMetering::Cheque with other classes

[mult from]	[mult to] name	type	description
[0..1]	[1..1] Tender	Tender	Payment tender the cheque is being used for.

6.10.19 ConsumptionTariffInterval root class

One of a sequence of intervals defined in terms of consumption quantity of a service such as electricity, water, gas, etc. It is typically used in association with TariffProfile to define the steps or blocks in a step tariff structure, where startValue simultaneously defines the entry value of this step and the closing value of the previous step. Where consumption is ≥ startValue it falls within this interval and where consumption is < startValue it falls within the previous interval.

Table 240 shows all attributes of ConsumptionTariffInterval.

Table 240 – Attributes of PaymentMetering::ConsumptionTariffInterval

name	type	description
sequenceNumber	Integer	A sequential reference that defines the identity of this interval and its relative position with respect to other intervals in a sequence of intervals.
startValue	RealEnergy	The lowest level of consumption that defines the starting point of this interval. The interval extends to the start of the next interval or until it is reset to the start of the first interval by TariffProfile.tariffCycle.

Table 241 shows all association ends of ConsumptionTariffInterval with other classes.

Table 241 – Association ends of PaymentMetering::ConsumptionTariffInterval with other classes

[mult from]	[mult to] name	type	description
[0..*]	[0..*] Charges	Charge	All charges used to define this consumption tariff interval.
[0..*]	[0..*] TouTariffIntervals	TimeTariffInterval	All time of use tariff intervals influenced by this consumption tariff interval.
[0..*]	[0..*] TariffProfiles	TariffProfile	All tariff profiles defined by this consumption tariff interval.

6.10.20 MerchantAccount

The operating account controlled by merchant agreement, against which the vendor may vend tokens or receipt payments. Transactions via vendor shift debit the account and bank deposits via bank statement credit the account.

Table 242 shows all attributes of MerchantAccount.

Table 242 – Attributes of PaymentMetering::MerchantAccount

name	type	description
currentBalance	Money	The current operating balance of this account.
provisionalBalance	Money	The balance of this account after taking into account any pending debits from VendorShift.merchantDebitAmount and pending credits from BankStatement.merchantCreditAmount or credits (see also BankStatement attributes and VendorShift attributes).
type	String	inherited from: Document
createdDateTime	DateTime	inherited from: Document
lastModifiedDateTime	DateTime	inherited from: Document
revisionNumber	String	inherited from: Document
electronicAddress	ElectronicAddress	inherited from: Document
subject	String	inherited from: Document
title	String	inherited from: Document
docStatus	Status	inherited from: Document
status	Status	inherited from: Document
aliasName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject

Table 243 shows all association ends of MerchantAccount with other classes.

Table 243 – Association ends of PaymentMetering::MerchantAccount with other classes

[mult from]	[mult to] name	type	description
[0..*]	[0..*] Transactors	Transactor	All transactors this merchant account is registered with.
[0..1]	[0..*] VendorShifts	VendorShift	All vendor shifts that operate on this merchant account.
[0..*]	[0..1] MerchantAgreement	MerchantAgreement	Merchant agreement that instantiated this merchant account.

[mult from]	[mult to] name	type	description
[0..1]	[0..*] ConfigurationEvents	ConfigurationEvent	inherited from: Document
[0..1]	[0..*] DiagramObjects	DiagramObject	inherited from: IdentifiedObject
[1..1]	[0..*] Names	Name	inherited from: IdentifiedObject

6.10.21 MerchantAgreement

A formal controlling contractual agreement between supplier and merchant, in terms of which the merchant is authorised to vend tokens and receipt payments on behalf of the supplier. The merchant is accountable to the supplier for revenue collected at point of sale.

Table 244 shows all attributes of MerchantAgreement.

Table 244 – Attributes of PaymentMetering::MerchantAgreement

name	type	description
signDate	Date	inherited from: Agreement
validityInterval	DateTimeInterval	inherited from: Agreement
type	String	inherited from: Document
createdDateTime	DateTime	inherited from: Document
lastModifiedDateTime	DateTime	inherited from: Document
revisionNumber	String	inherited from: Document
electronicAddress	ElectronicAddress	inherited from: Document
subject	String	inherited from: Document
title	String	inherited from: Document
docStatus	Status	inherited from: Document
status	Status	inherited from: Document
aliasName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject

Table 245 shows all association ends of MerchantAgreement with other classes.

Table 245 – Association ends of PaymentMetering::MerchantAgreement with other classes

[mult from]	[mult to] name	type	description
[0..1]	[0..*] MerchantAccounts	MerchantAccount	All merchant accounts instantiated as a result of this merchant agreement.
[0..1]	[0..*] ConfigurationEvents	ConfigurationEvent	inherited from: Document
[0..1]	[0..*] DiagramObjects	DiagramObject	inherited from: IdentifiedObject
[1..1]	[0..*] Names	Name	inherited from: IdentifiedObject

6.10.22 PointOfSale

Logical point where transactions take place with operational interaction between cashier and the payment system; in certain cases the point of sale interacts directly with the end

customer, in which case the cashier might not be a real person: for example a self-service kiosk or over the internet.

Table 246 shows all attributes of PointOfSale.

Table 246 – Attributes of PaymentMetering::PointOfSale

name	type	description
location	String	Local description for where this point of sale is physically located.
aliasName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject

Table 247 shows all association ends of PointOfSale with other classes.

Table 247 – Association ends of PaymentMetering::PointOfSale with other classes

[mult from]	[mult to] name	type	description
[0..1]	[0..*] CashierShifts	CashierShift	All shifts this point of sale operated in.
[0..1]	[0..*] DiagramObjects	DiagramObject	inherited from: IdentifiedObject
[1..1]	[0..*] Names	Name	inherited from: IdentifiedObject

6.10.23 Receipt

Record of total receipted payment from customer.

Table 248 shows all attributes of Receipt.

Table 248 – Attributes of PaymentMetering::Receipt

name	type	description
isBankable	Boolean	True if this receipted payment is manually bankable, otherwise it is an electronic funds transfer.
line	LineDetail	Receipted amount with rounding, date and note.
aliasName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject

Table 249 shows all association ends of Receipt with other classes.

Table 249 – Association ends of PaymentMetering::Receipt with other classes

[mult from]	[mult to] name	type	description
[0..1]	[1..*] Transactions	Transaction	All transactions recorded for this receipted payment.
[0..*]	[0..1] CashierShift	CashierShift	Cashier shift during which this receipt was recorded.
[0..*]	[0..1] VendorShift	VendorShift	Vendor shift during which this receipt was recorded.

[mult from]	[mult to] name	type	description
[1..1]	[1..*] Tenders	Tender	All payments received in the form of tenders recorded by this receipt.
[0..1]	[0..*] DiagramObjects	DiagramObject	inherited from: IdentifiedObject
[1..1]	[0..*] Names	Name	inherited from: IdentifiedObject

6.10.24 ServiceSupplier

Organisation that provides services to customers.

Table 250 shows all attributes of ServiceSupplier.

Table 250 – Attributes of PaymentMetering::ServiceSupplier

name	type	description
kind	SupplierKind	Kind of supplier.
issuerIdentificationNumber	String	Unique transaction reference prefix number issued to an entity by the International Organization for Standardization for the purpose of tagging onto electronic financial transactions, as defined in ISO/IEC 7812-1 and ISO/IEC 7812-2.
aliasName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject

Table 251 shows all association ends of ServiceSupplier with other classes.

Table 251 – Association ends of PaymentMetering::ServiceSupplier with other classes

[mult from]	[mult to] name	type	description
[1..1]	[0..*] CustomerAgreements	CustomerAgreement	All customer agreements of this service supplier.
[0..1]	[0..*] UsagePoints	UsagePoint	All usage points this service supplier utilises to deliver a service.
[0..1]	[0..*] ConfigurationEvents	ConfigurationEvent	inherited from: OrganisationRole
[0..*]	[0..1] Organisation	Organisation	inherited from: OrganisationRole
[0..1]	[0..*] DiagramObjects	DiagramObject	inherited from: IdentifiedObject
[1..1]	[0..*] Names	Name	inherited from: IdentifiedObject

6.10.25 Shift

Generally referring to a period of operation or work performed. Whether the shift is open/closed can be derived from attributes 'activityInterval.start' and 'activityInterval.end'.

The grand total for receipts (i.e., cumulative total of all actual receipted amounts during this shift; bankable + non-bankable; excludes rounding error totals) can be derived from receipt:

=sum(Receipt.receiptAmount); includes bankable and non-bankable receipts.

It also has to be reconciled against:

=sum(receiptsGrandTotalBankable + receiptsGrandTotalNonBankable).

It also has to be reconciled against ReceiptSummary:

=sum(ReceiptSummary.receiptsTotal).

The attributes with "GrandTotal" defined in this class may need to be used when the source data is periodically flushed from the system and then these cannot be derived.

Table 252 shows all attributes of Shift.

Table 252 – Attributes of PaymentMetering::Shift

name	type	description
activityInterval	DateTimeInterval	Interval for activity of this shift.
receiptsGrandTotalBankable	Money	Total of amounts receipted during this shift that can be manually banked (cash and cheques for example). Values are obtained from Receipt attributes: =sum(Receipt.receiptAmount) for all Receipt.bankable = true.
receiptsGrandTotalNonBankable	Money	Total of amounts receipted during this shift that cannot be manually banked (card payments for example). Values are obtained from Receipt attributes: =sum(Receipt.receiptAmount) for all Receipt.bankable = false.
receiptsGrandTotalRounding	Money	Cumulative amount in error due to process rounding not reflected in receiptsGrandTotal. Values are obtained from Receipt attributes: =sum(Receipt.receiptRounding).
transactionsGrandTotal	Money	Cumulative total of transacted amounts during this shift. Values are obtained from Transaction attributes: =sum(Transaction.transactionAmount). It also has to be reconciled against TransactionSummary: =sum(TransactionSummary.transactionsTotal).
transactionsGrandTotalRounding	Money	Cumulative amount in error due to process rounding not reflected in transactionsGandTotal. Values are obtained from Transaction attributes: =sum(Transaction.transactionRounding).
aliasName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject

Table 253 shows all association ends of Shift with other classes.

Table 253 – Association ends of PaymentMetering::Shift with other classes

[mult from]	[mult to] name	type	description
[0..1]	[0..*] DiagramObjects	DiagramObject	inherited from: IdentifiedObject
[1..1]	[0..*] Names	Name	inherited from: IdentifiedObject

6.10.26 TariffProfile

A schedule of charges; structure associated with Tariff that allows the definition of complex tarif structures such as step and time of use when used in conjunction with TimeTariffInterval

and Charge. Inherited 'status.value' is defined in the context of the utility's business rules, for example: active, inactive, etc.

Table 254 shows all attributes of TariffProfile.

Table 254 – Attributes of PaymentMetering::TariffProfile

name	type	description
tariffCycle	String	The frequency at which the tariff charge schedule is repeated. Examples are: once off on a specified date and time; hourly; daily; weekly; monthly; 3-monthly; 6-monthly; 12-monthly; etc. At the end of each cycle, the business rules are reset to start from the beginning again.
type	String	inherited from: Document
createdDateTime	DateTime	inherited from: Document
lastModifiedDateTime	DateTime	inherited from: Document
revisionNumber	String	inherited from: Document
electronicAddress	ElectronicAddress	inherited from: Document
subject	String	inherited from: Document
title	String	inherited from: Document
docStatus	Status	inherited from: Document
status	Status	inherited from: Document
aliasName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject

Table 255 shows all association ends of TariffProfile with other classes.

Table 255 – Association ends of PaymentMetering::TariffProfile with other classes

[mult from]	[mult to] name	type	description
[0..*]	[0..*] ConsumptionTariffIntervals	ConsumptionTariffInterval	All consumption tariff intervals used to define this tariff profile.
[0..*]	[0..*] TimeTariffIntervals	TimeTariffInterval	All time tariff intervals used to define this tariff profile.
[0..*]	[0..*] Tariffs	Tariff	All tariffs defined by this tariff profile.
[0..1]	[0..*] ConfigurationEvents	ConfigurationEvent	inherited from: Document
[0..1]	[0..*] DiagramObjects	DiagramObject	inherited from: IdentifiedObject
[1..1]	[0..*] Names	Name	inherited from: IdentifiedObject

6.10.27 Tender

Tender is what is "offered" by the customer towards making a payment and is often more than the required payment (hence the need for 'change'). The payment is thus that part of the Tender that goes towards settlement of a particular transaction.

Tender is modelled as an aggregation of Cheque and Card. Both these tender types can exist in a single tender bid thus 'accountHolderName' has to exist separately in each of Cheque and Card as each could have a different account holder name.

Table 256 shows all attributes of Tender.

Table 256 – Attributes of PaymentMetering::Tender

name	type	description
kind	TenderKind	Kind of tender from customer.
amount	Money	Amount tendered by customer.
change	Money	Difference between amount tendered by customer and the amount charged by point of sale.
aliasName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject

Table 257 shows all association ends of Tender with other classes.

Table 257 – Association ends of PaymentMetering::Tender with other classes

[mult from]	[mult to] name	type	description
[1..*]	[1..1] Receipt	Receipt	Receipt that recorded this receiving of a payment in the form of tenders.
[1..1]	[0..1] Card	Card	Card used to tender payment.
[1..1]	[0..1] Cheque	Cheque	Cheque used to tender payment.
[0..1]	[0..*] DiagramObjects	DiagramObject	inherited from: IdentifiedObject
[1..1]	[0..*] Names	Name	inherited from: IdentifiedObject

6.10.28 TimeTariffInterval root class

One of a sequence of time intervals defined in terms of real time. It is typically used in association with TariffProfile to define the intervals in a time of use tariff structure, where startDateTime simultaneously determines the starting point of this interval and the ending point of the previous interval.

Table 258 shows all attributes of TimeTariffInterval.

Table 258 – Attributes of PaymentMetering::TimeTariffInterval

name	type	description
sequenceNumber	Integer	A sequential reference that defines the identity of this interval and its relative position with respect to other intervals in a sequence of intervals.
startTime	Time	A real time marker that defines the starting time (typically it is the time of day) for this interval. The interval extends to the start of the next interval or until it is reset to the start of the first interval by TariffProfile.tariffCycle.

Table 259 shows all association ends of TimeTariffInterval with other classes.

Table 259 – Association ends of PaymentMetering::TimeTariffInterval with other classes

[mult from]	[mult to] name	type	description
[0..*]	[0..*] Charges	Charge	All charges used to define this time tariff interval.
[0..*]	[0..*] ConsumptionTariffIntervals	ConsumptionTariffInterval	All consumption tariff intervals that introduce variation in this time of use tariff interval; allows to express e.g., peak hour prices that are different with different consumption blocks.
[0..*]	[0..*] TariffProfiles	TariffProfile	All tariff profiles defined by this time tariff interval.

6.10.29 Transaction

The record of details of payment for service or token sale.

Table 260 shows all attributes of Transaction.

Table 260 – Attributes of PaymentMetering::Transaction

name	type	description
kind	TransactionKind	Kind of transaction.
receiverReference	String	Reference to the entity that is the recipient of 'amount' (for example, supplier for service charge payment; or tax receiver for VAT).
donorReference	String	Reference to the entity that is the source of 'amount' (for example: customer for token purchase; or supplier for free issue token).
diverseReference	String	Formal reference for use with diverse payment (traffic fine for example).
reversedId	String	(if 'kind' is transactionReversal) Reference to the original transaction that is being reversed by this transaction.
serviceUnitsEnergy	RealEnergy	Actual amount of service units that is being paid for.
serviceUnitsError	RealEnergy	Number of service units not reflected in 'serviceUnitsEnergy' due to process rounding or truncating errors.
line	LineDetail	Transaction amount, rounding, date and note for this transaction line.
aliasName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject

Table 261 shows all association ends of Transaction with other classes.

Table 261 – Association ends of PaymentMetering::Transaction with other classes

[mult from]	[mult to] name	type	description
[0..*]	[0..1] Meter	Meter	Meter for this vending transaction.
[0..1]	[0..*] UserAttributes	UserAttribute	All snapshots of meter parameters recorded at the time of this transaction. Use 'name' and 'value.value' attributes to specify name and value of a parameter from meter.
[0..*]	[0..1] CustomerAccount	CustomerAccount	Customer account for this payment transaction.
[0..*]	[0..1] PricingStructure	PricingStructure	Pricing structure applicable for this transaction.
[0..*]	[0..1] AuxiliaryAccount	AuxiliaryAccount	Auxiliary account for this payment transaction.
[0..*]	[0..1] CashierShift	CashierShift	Cashier shift during which this transaction was recorded.
[1..*]	[0..1] Receipt	Receipt	The receipted payment for which this transaction has been recorded.
[0..*]	[0..1] VendorShift	VendorShift	Vendor shift during which this transaction was recorded.
[0..1]	[0..*] DiagramObjects	DiagramObject	inherited from: IdentifiedObject
[1..1]	[0..*] Names	Name	inherited from: IdentifiedObject

6.10.30 Transactor

The entity that ultimately executes the transaction and which is in control of the process; typically this is embodied in secure software running on a server that may employ secure hardware encryption devices for secure transaction processing.

Table 262 shows all attributes of Transactor.

Table 262 – Attributes of PaymentMetering::Transactor

name	type	description
aliasName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject

Table 263 shows all association ends of Transactor with other classes.

Table 263 – Association ends of PaymentMetering::Transactor with other classes

[mult from]	[mult to] name	type	description
[0..*]	[0..*] MerchantAccounts	MerchantAccount	All merchant accounts registered with this transactor.
[0..1]	[0..*] DiagramObjects	DiagramObject	inherited from: IdentifiedObject
[1..1]	[0..*] Names	Name	inherited from: IdentifiedObject

6.10.31 Vendor

The entity that owns the point of sale and contracts with the cashier to receipt payments and vend tokens using the payment system. The vendor has a private contract with and is managed by the merchant which is a type of organisation. The vendor is accountable to the merchant for revenue collected, and the merchant is in turn accountable to the supplier.

Table 264 shows all attributes of Vendor.

Table 264 – Attributes of PaymentMetering::Vendor

name	type	description
aliasName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject

Table 265 shows all association ends of Vendor with other classes.

Table 265 – Association ends of PaymentMetering::Vendor with other classes

[mult from]	[mult to] name	type	description
[0..1]	[0..*] VendorShifts	VendorShift	All vendor shifts opened and owned by this vendor.
[0..1]	[0..*] DiagramObjects	DiagramObject	inherited from: IdentifiedObject
[1..1]	[0..*] Names	Name	inherited from: IdentifiedObject

6.10.32 VendorShift

The operating shift for a vendor during which the vendor may transact against the merchant's account. It aggregates transactions and receipts during the shift and periodically debits a merchant account. The totals in VendorShift should always = sum of totals aggregated in all cashier shifts that were open under the particular vendor shift.

Table 266 shows all attributes of VendorShift.

Table 266 – Attributes of PaymentMetering::VendorShift

name	type	description
merchantDebitAmount	Money	The amount that is to be debited from the merchant account for this vendor shift. This amount reflects the sum(PaymentTransaction.transactionAmount).
posted	Boolean	If true, merchantDebitAmount has been debited from MerchantAccount; typically happens at the end of VendorShift when it closes.
activityInterval	DateTimeInterval	inherited from: Shift
receiptsGrandTotalBankable	Money	inherited from: Shift
receiptsGrandTotalNonBankable	Money	inherited from: Shift
receiptsGrandTotalRounding	Money	inherited from: Shift
transactionsGrandTotal	Money	inherited from: Shift
transactionsGrandTotalRounding	Money	inherited from: Shift
aliasName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject

Table 267 shows all association ends of VendorShift with other classes.

Table 267 – Association ends of PaymentMetering::VendorShift with other classes

[mult from]	[mult to] name	type	description
[0..1]	[0..*] Receipts	Receipt	All receipts recorded during this vendor shift.
[0..1]	[0..*] Transactions	Transaction	All transactions recorded during this vendor shift.
[0..*]	[0..1] MerchantAccount	MerchantAccount	Merchant account this vendor shift periodically debits (based on aggregated transactions).
[0..*]	[0..1] Vendor	Vendor	Vendor that opens and owns this vendor shift.
[0..1]	[0..*] DiagramObjects	DiagramObject	inherited from: IdentifiedObject
[1..1]	[0..*] Names	Name	inherited from: IdentifiedObject

Bibliography

IEC 60050 (all parts), *International Electrotechnical vocabulary* (available at http://www.electropedia.org)

IEC 61968-3, *Application integration at electric utilities – System interfaces for distribution management – Part 3: Interface for network operations*

IEC 61968-4, *Application integration at electric utilities – System interfaces for distribution management – Part 4: Interfaces for records and asset management*

IEC 61968-5, *Application integration at electric utilities – System interfaces for distribution management – Part 5: Interfaces for operational planning and optimisation*[16]

IEC 61968-6, *Application integration at electric utilities – System interfaces for distribution management – Part 6: Interfaces for maintenance and construction*[17]

IEC 61968-7, *Application integration at electric utilities – System interfaces for distribution management – Part 7: Interfaces for network expansion planning*[18]

IEC 61968-8, *Application integration at electric utilities – System interfaces for distribution management – Part 8: Interfaces for customer support*[19]

IEC 61968-9, *Application integration at electric utilities – System interfaces for distribution management – Part 9: Interfaces for meter reading and control*

IEC 61968-13, *Application integration at electric utilities – System interfaces for distribution management – Part 13: CIM RDF Model exchange format for distribution*

IEC 61968-100, *Application integration at electric utilities – System interfaces for distribution management – Part 100: Implementation Profiles for IEC 61968*[20]

IEC 61970-552, *Energy management system application program interfaces – Part 552: CIM XML Model Exchange Format*[21]

IEC 62325-301, *Framework for energy market communications – Part 301: Common Information Model (CIM) extensions for markets*

ISO/IEC 7812-1, *Identification cards – Identification of issuers – Part 1: Numbering system*

ISO/IEC 7812-2, *Identification cards – Identification of issuers – Part 2: Application and registration procedures*

ANSI C12.10, *American National Standard for Physical Aspects of Watthour Meters – Safety Standard*

ANSI C12.18, *American National Standard for Protocol Specification for ANSI Type 2 Optical Port*

---

16 Under consideration.

17 Under consideration.

18 Under consideration.

19 Under consideration.

20 Under consideration.

21 Under consideration.

SOMMAIRE

COMMISSION ÉLECTROTECHNIQUE INTERNATIONALE

INTÉGRATION D'APPLICATIONS POUR LES SERVICES ÉLECTRIQUES – INTERFACES SYSTÈME POUR LA GESTION DE DISTRIBUTION –

Partie 11: Extensions du modèle d'information commun (CIM) pour la distribution

AVANT-PROPOS

1) La Commission Electrotechnique Internationale (CEI) est une organisation mondiale de normalisation composée de l'ensemble des comités électrotechniques nationaux (Comités nationaux de la CEI). La CEI a pour objet de favoriser la coopération internationale pour toutes les questions de normalisation dans les domaines de l'électricité et de l'électronique. A cet effet, la CEI – entre autres activités – publie des Normes internationales, des Spécifications techniques, des Rapports techniques, des Spécifications accessibles au public (PAS) et des Guides (ci-après dénommés "Publication(s) de la CEI"). Leur élaboration est confiée à des comités d'études, aux travaux desquels tout Comité national intéressé par le sujet traité peut participer. Les organisations internationales, gouvernementales et non gouvernementales, en liaison avec la CEI, participent également aux travaux. La CEI collabore étroitement avec l'Organisation Internationale de Normalisation (ISO), selon des conditions fixées par accord entre les deux organisations.
2) Les décisions ou accords officiels de la CEI concernant les questions techniques représentent, dans la mesure du possible, un accord international sur les sujets étudiés, étant donné que les Comités nationaux de la CEI intéressés sont représentés dans chaque comité d'études.
3) Les Publications de la CEI se présentent sous la forme de recommandations internationales et sont agréées comme telles par les Comités nationaux de la CEI. Tous les efforts raisonnables sont entrepris afin que la CEI s'assure de l'exactitude du contenu technique de ses publications; la CEI ne peut pas être tenue responsable de l'éventuelle mauvaise utilisation ou interprétation qui en est faite par un quelconque utilisateur final.
4) Dans le but d'encourager l'uniformité internationale, les Comités nationaux de la CEI s'engagent, dans toute la mesure possible, à appliquer de façon transparente les Publications de la CEI dans leurs publications nationales et régionales. Toutes divergences entre toutes Publications de la CEI et toutes publications nationales ou régionales correspondantes doivent être indiquées en termes clairs dans ces dernières.
5) La CEI elle-même ne fournit aucune attestation de conformité. Des organismes de certification indépendants fournissent des services d'évaluation de conformité et, dans certains secteurs, accèdent aux marques de conformité de la CEI. La CEI n'est responsable d'aucun des services effectués par les organismes de certification indépendants.
6) Tous les utilisateurs doivent s'assurer qu'ils sont en possession de la dernière édition de cette publication.
7) Aucune responsabilité ne doit être imputée à la CEI, à ses administrateurs, employés, auxiliaires ou mandataires, y compris ses experts particuliers et les membres de ses comités d'études et des Comités nationaux de la CEI, pour tout préjudice causé en cas de dommages corporels et matériels, ou de tout autre dommage de quelque nature que ce soit, directe ou indirecte, ou pour supporter les coûts (y compris les frais de justice) et les dépenses découlant de la publication ou de l'utilisation de cette Publication de la CEI ou de toute autre Publication de la CEI, ou au crédit qui lui est accordé.
8) L'attention est attirée sur les références normatives citées dans cette publication. L'utilisation de publications référencées est obligatoire pour une application correcte de la présente publication.
9) L'attention est attirée sur le fait que certains des éléments de la présente Publication de la CEI peuvent faire l'objet de droits de brevet. La CEI ne saurait être tenue pour responsable de ne pas avoir identifié de tels droits de brevets et de ne pas avoir signalé leur existence.

La Norme internationale CEI 61968-11 a été établie par le comité d'études 57 de la CEI: Gestion des systèmes de puissance et échanges d'informations associés.

Le texte de cette norme est issu des documents suivants:

FDIS	Rapport de vote
57/1295/FDIS	57/1326/RVD

Le rapport de vote indiqué dans le tableau ci-dessus donne toute information sur le vote ayant abouti à l'approbation de cette norme.

Cette publication a été rédigée selon les Directives ISO/CEI, Partie 2.

Cette deuxième édition annule et remplace la première édition parue en 2010. Elle constitue une révision technique.

Les principaux changements par rapport à la première édition sont résumés ci-dessous[1];

- Introduction de nouvelles classes pour prendre en charge la dénomination de manière flexible des "identified objects" (nouvelles classes disponibles dans le CIM de base, CEI 61970-301).
- Introduction de nouvelles classes pour prendre en charge l'échange de schémas unifilaires (nouvelles classes disponibles dans le CIM de base, CEI 61970-301).
- Modèles de transport et de distribution consolidés pour les lignes, transformateurs, organes de commutation, organes de détection et autres équipements auxiliaires (certaines classes de l'Ed.1 ont été légèrement modifiées et déplacées du DCIM dans le CIM de base, CEI 61970-301, autres nouvelles classes disponibles dans le CIM de base, CEI 61970-301).
- Prise en charge des définitions de phases séparées, généralement nécessaires pour la modélisation de réseau déséquilibrés (nouvelles classes disponibles dans le CIM de base, CEI 61970-301).
- Prise en charge des changements de réseau temporaires au moyen de modèles de coupures, cavaliers et colliers (nouvelles classes disponibles dans le CIM de base, CEI 61970-301).
- Modèle flexible applicable aux organisations et à leurs rôles.
- Prise en charge des systèmes de coordonnées dans la description des emplacements géographiques.
- Prise en charge du suivi des événements de configuration.
- Modèle allégé applicable aux biens (assets) et catalogues de biens.
- Prise en charge du lien entre les modèles orientés réseau et les modèles orientés locaux (comptage).
- Prise en charge des dispositifs de réseau dans un local.

Dans les sections informatives du présent document, les termes en caractère Arial Noir s'appliquent aux termes utilisés comme des jetons dans les articles normatifs (pour faciliter la lecture et la recherche documentaire).

Une liste de toutes les parties de la série CEI 61968, sous le titre général: *Intégration d'applications pour les services électriques – Interfaces système pour la gestion de distribution*, peut être consultée sur le site web de la CEI.

1 Pour les améliorations apportées au CIM de base, voir la CEI 61970-301 documentant le CIM15.

Le comité a décidé que le contenu de cette publication ne sera pas modifié avant la date de stabilité indiquée sur le site web de la CEI sous "http://webstore.iec.ch" dans les données relatives à la publication recherchée. A cette date, la publication sera

- reconduite,
- supprimée,
- remplacée par une édition révisée, ou
- amendée.

IMPORTANT – Le logo "colour inside" qui se trouve sur la page de couverture de cette publication indique qu'elle contient des couleurs qui sont considérées comme utiles à une bonne compréhension de son contenu. Les utilisateurs devraient, par conséquent, imprimer cette publication en utilisant une imprimante couleur.

INTRODUCTION

La série des normes CEI 61968 est prévue pour faciliter l'intégration inter-applications, par opposition à l'intégration intra-applications. L'intégration intra-applications est destinée aux programmes d'un même système, communiquant habituellement les uns avec les autres en utilisant des intergiciels (middleware) qui sont intégrés dans leur environnement d'exécution sous-jacent et tendent à être optimisés pour des connexions proches, en temps réel et synchrones, et des interrogations/réponses interactives ou des modèles de communication conversationnels. Par conséquent, ces normes d'interface inter-applications sont appropriées pour les applications faiblement couplées avec une plus grande hétérogénéité dans le langage, les systèmes d'exploitation, les protocoles et des outils de gestion. Cette série de normes est prévue pour supporter des applications qui nécessitent l'échange de données environ toutes les secondes, minutes ou heures, plutôt que d'attendre un traitement de nuit par lot. Cette série de normes, qui est destinée à être mise en œuvre avec des services d'intergiciel, qui échangent des messages parmi des applications, complétera, ne remplacera pas, les entrepôts de données de l'entreprise de distribution, les passerelles de base de données, et les archives opérationnelles.

Au sens de la CEI 61968, un système de gestion de distribution (DMS – distribution management system) se compose de divers composants d'application distribués permettant à l'entreprise de distribution de gérer les réseaux de distribution électriques. Ces capacités incluent la surveillance et la commande des équipements de fourniture d'énergie, les processus de gestion qui assurent la fiabilité du système, la gestion de la tension, la gestion de la demande, la gestion des interruptions de service, la gestion des travaux, la cartographie automatisée et la gestion des équipements. Des interfaces normalisées sont définies pour chaque classe d'applications identifiée dans le modèle d'interface de référence (IRM – interface reference model), qui est décrit dans la CEI 61968-1.

INTÉGRATION D'APPLICATIONS POUR LES SERVICES ÉLECTRIQUES – INTERFACES SYSTÈME POUR LA GESTION DE DISTRIBUTION –

Partie 11: Extensions du modèle d'information commun (CIM) pour la distribution

1 Domaine d'application

La présente partie de la CEI 61968 spécifie les extensions pour la distribution du modèle d'information commun (CIM – common information model) spécifié dans la CEI 61970-301. Elle définit un ensemble normalisé d'extensions du modèle d'information commun (CIM), qui prennent en charge les définitions de message données dans la CEI 61968-3 à la CEI 61968-9, la CEI 61968-13 et la CEI 61968-14[2]. Le domaine d'application de la présente norme est le modèle d'information qui étend le CIM de base pour les besoins des réseaux de distribution, ainsi que pour l'intégration avec les systèmes d'information à l'échelle de l'entreprise qui sont typiquement utilisés dans les entreprises de distribution électrique. Le modèle d'information est défini en UML, langage indépendant de la plate-forme et pouvant être traité électroniquement, qui est alors utilisé pour créer les définitions de la charge utile de message dans différents formats requis. Ainsi, la présente norme ne subira pas l'impact de la spécification, du développement et/ou du déploiement d'infrastructures de prochaine génération, ni par l'utilisation de normes ni par des moyens propriétaires.

Pour les besoins de la présente partie de la CEI 61968, le modèle de distribution CIM (DCIM) se réfère au modèle CIM de la CEI tel que défini dans la CEI 61970-301 et la présente partie de la CEI 61968.

Le modèle d'information commun (CIM) est un modèle abstrait de tous les objets principaux d'une entreprise de service public de distribution d'électricité habituellement impliqués dans les opérations de l'entreprise. En fournissant une façon normalisée de représenter les ressources des réseaux électriques comme classes et attributs d'objets, ainsi que leurs relations, le CIM facilite l'intégration des applications logicielles développées de façon indépendante par différents fournisseurs. Le CIM facilite l'intégration en définissant un langage commun (c'est-à-dire une sémantique et une syntaxe) fondé sur le modèle CIM pour permettre à ces applications ou systèmes d'accéder aux données publiques et d'échanger des informations indépendamment de la représentation interne de ces informations.

La CEI 61970-301 définit un noyau CIM pour les applications de système de gestion de l'énergie (EMS – *energy management system*), y compris de nombreuses classes qui seraient utiles dans une plus grande diversité d'applications. En raison de sa taille, les classes du CIM sont groupées en paquetages logiques, et les collections de ces paquetages sont maintenues sous forme de Normes internationales distinctes. Le présent document étend le noyau CIM par des paquetages qui sont axés sur les systèmes de gestion de distribution (DMS – distribution management system) comprenant les Assets (c'est-à-dire les biens), Work (c'est-à-dire les travaux), Customers (c'est-à-dire les clients), Load control (c'est-à-dire le contrôle de la charge), Metering (c'est-à-dire le comptage) et autres. La CEI 62325-301[3] étend le CIM par des paquetages qui sont axés sur les applications d'opérations (Market operations) et de régulation du marché (Market management). D'autres extensions du CIM peuvent être publiées sous forme de Normes internationales, qui sont chacune maintenues par un groupe distinct d'experts du domaine. En fonction des besoins d'un projet, l'intégration d'applications peut exiger des classes et des paquetages issus d'une ou de plusieurs normes du CIM.

---

2 Les CEI 61968-5, CEI 61968-6, CEI 61968-7, CEI 61968-8 et CEI 61968-14 sont à l'étude.

3 A l'étude.

2 Références normatives

Les documents suivants sont cités en référence de manière normative, en intégralité ou en partie, dans le présent document et sont indispensables pour son application. Pour les références datées, seule l'édition citée s'applique. Pour les références non datées, la dernière édition du document de référence s'applique (y compris les éventuels amendements).

CEI 60076-1, *Transformateurs de puissance – Partie 1: Généralités*

CEI 61968-1, *Intégration d'applications pour les services électriques – Interfaces système pour la gestion de distribution – Partie 1: Architecture des interfaces et recommandations générales*

CEI 61968-2, *Application integration at electric utilities – System interfaces for distribution management – Part 2: Glossary* (disponible en anglais uniquement)

CEI 61970-301, *Interface de programmation d'application pour système de gestion d'énergie (EMS-API) – Partie 301: Base de modèle d'information commun (CIM)*[4]

CEI 61970-501, *Energy management system application program interface (EMS-API) – Part 501: Common Information Model Resource Description Framework (CIM RDF) schema* (disponible en anglais uniquement) (disponible en anglais uniquement)

CEI 62361-100, *Naming and Design Rules for CIM Profiles to XML Schema Mapping*[5] (disponible en anglais uniquement)

IEEE 802.3, *IEEE Standard for Information technology-Specific requirements – Part 3: Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications*

3 Termes et définitions

Pour les besoins du présent document, les termes et définitions donnés dans la CEI 61968-2 et les suivants s'appliquent.

NOTE Se référer à la CEI 60050, Vocabulaire Électrotechnique International, pour les définitions du glossaire général.

3.1
système de gestion d'énergie
EMS (en anglais *energy management system*)
système informatique comprenant une plate-forme logicielle offrant les services de support de base et un ensemble d'applications offrant les fonctionnalités requises pour le bon fonctionnement des installations de production et de transmission d'électricité afin d'assurer la sécurité adéquate d'approvisionnement énergétique à un coût minimal

3.2
système de gestion de distribution
DMS (en anglais *distribution management system*)
système informatique comprenant une plate-forme logicielle offrant les services de support de base et un ensemble d'applications offrant les fonctionnalités requises pour le bon fonctionnement des installations de distribution d'électricité afin d'assurer la sécurité adéquate d'approvisionnement énergétique à un coût minimal

---

4 5e édition. A l'étude.

5 A l'étude.

3.3
langage de modélisation unifié
UML (en anglais *unified modeling language*)
langage descriptif formel et complet avec des techniques de présentation de diagrammes utilisées pour représenter des systèmes logiciels, depuis l'analyse des exigences, en passant par la conception et la mise en œuvre, jusqu'à la documentation

Note 1 à l'article: Le langage UML a évolué pour devenir une Norme internationale en passant d'une collection de méthodes apportées par différents pratiquants. Pour définir le modèle, le CIM s'appuie sur le langage UML à partir duquel des outils automatisés génèrent directement la documentation, les schémas et autres objets d'art. Une compréhension de base du langage UML est nécessaire pour comprendre le CIM.

3.4
modèle d'information commun avec extensions pour la distribution
DCIM (en anglais *common information model with distribution extensions*)
union du CIM central selon la CEI 61970-301 et de paquetages complémentaires définis dans le présent document, la CEI 61968-11

Note 1 à l'article: Le DCIM vise à satisfaire à la plupart des besoins de modélisation de domaine d'un DMS, mais un projet spécifique peut exiger d'autres paquetages ou extensions du CIM.

3.5
profil
sous-ensemble des classes, associations et attributs du DCIM nécessaires pour accomplir un type spécifique d'interface

Note 1 à l'article: Il peut être exprimé dans des fichiers XSD, RDF et/ou OWL. Un profil peut être soumis à essai entre des applications. Un profil est nécessaire pour "utiliser" le DCIM. Plusieurs profils sont définis dans d'autres parties de la famille de normes CEI 61968.

3.6
schéma XML
schéma utilisé pour définir la structure, le contenu et la sémantique de fichiers XML (eXtensible Markup Language, c'est-à-dire langage de balisage extensible)

Note 1 à l'article: Les schémas XML se trouvent généralement dans des fichiers ayant une extension «xsd». Le DCIM utilise des fichiers XSD pour définir des messages inter-applications dans la plupart des domaines, excepté l'échange de modèles de réseau électrique.

3.7
cadre de description des ressources
RDF (en anglais *resource description framework*)
norme web (W3C) utilisée pour représenter les modèles d'informations

Note 1 à l'article: RDF est plus puissant que XSD car il peut décrire un modèle de données, pas seulement un fichier XML. Le DCIM utilise un sous-ensemble du RDF pour prendre en charge l'échange de modèles de réseau électrique.

3.8
langage d'ontologies web
OWL (en anglais *web ontology language*)
autre norme Web (W3C) qui inclut RDF et l'étend

Note 1 à l'article: Le langage OWL est plus puissant que RDF pour la prise en charge des types de données, des énumérations, davantage de détails des relations et associations de classes, etc. Les futurs profils DCIM peuvent utiliser le langage OWL.

4 Spécification CIM

4.1 Notation de modélisation du CIM

Le CIM est défini en utilisant des techniques de modélisation orientées objet. Plus précisément, la spécification du CIM utilise la notation du Langage de modélisation unifié (UML – Unified modeling language) qui définit le CIM comme un groupe de paquetages.

Chaque paquetage du CIM contient un ou plusieurs diagrammes de classe montrant sous forme graphique toutes les classes de ce paquetage et leurs relations. Chaque classe est ensuite définie de façon textuelle en mettant l'accent sur ses attributs et ses relations avec d'autres classes.

La notation UML est décrite dans les documents de l'Object Management Group (OMG) et dans plusieurs autres manuels édités.

4.2 Paquetages CIM

4.2.1 Généralités

Le CIM est fractionné en un ensemble de paquetages. Un paquetage est un moyen général permettant de grouper des éléments liés à un modèle. Les paquetages ont été choisis pour simplifier la conception, la compréhension et la révision du modèle. Le CIM se compose de plusieurs ensembles de paquetages. Des entités peuvent avoir des associations dépassant les limites de plusieurs paquetages. Chaque application utilisera des informations représentées dans plusieurs paquetages.

4.2.2 Paquetages CIM

Le CIM complet est réparti en groupes de paquetages pour des raisons de commodité. Les groupes comportant un DCIM comprennent:

- la CEI 61970-301 (CIM de base, définissant les types de données et les ressources des réseaux électriques tels que requis par les applications relatives aux centres de commande des systèmes EMS et DMS);
- la présente partie de la CEI 61968.

La Figure 1 montre les paquetages normatifs CIM de la CEI actuellement applicables[6] et leurs relations de dépendance. Les lignes tiretées indiquent une relation de dépendance, la flèche allant du paquetage dépendant vers le paquetage dont il dépend.

6 Les extensions CIM pour les marchés, CEI 62325-301 sont également les paquetages normatifs CIM, mais ne sont pas utilisées dans le DCIM et de ce fait ne sont pas affichées.

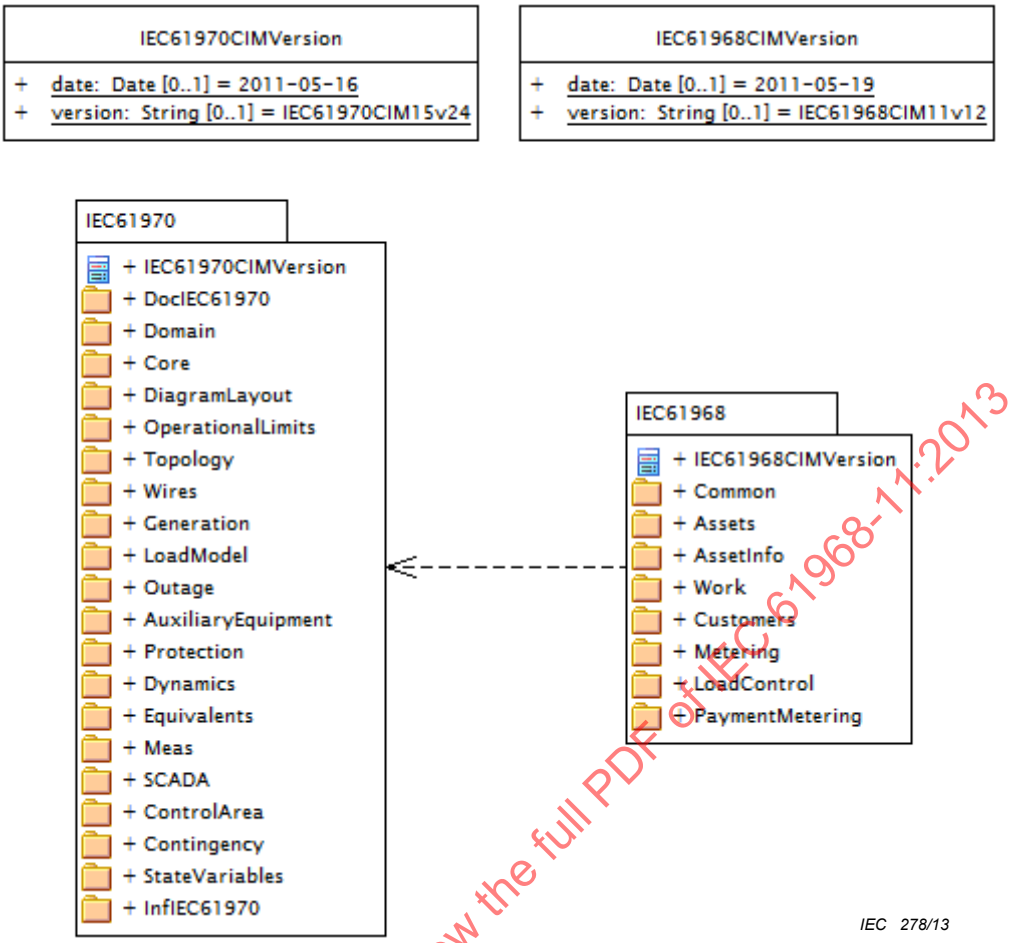


Figure 1 – Paquetages CIM

NOTE Le contenu du CIM de base auquel il est fait référence issu de la présente partie de la CEI 61968 a été généré automatiquement à partir de la version de modèle électronique UML du CIM de base IEC61970CIM15v31.

4.2.3 Paquetages des extensions du CIM pour la distribution (le présent document)

Le modèle du CIM de base tel que défini par la CEI 61970-301 définit un ensemble de sous-paquetages qui inclut **Domain** (c'est-à-dire le domaine), **Wires** (c'est-à-dire les fils), **AuxiliaryEquipment** (c'est-à-dire les équipements auxiliaires), **Topology** (c'est-à-dire la topologie), **Measurements** (c'est-à-dire les mesures), **Equivalents** (c'est-à-dire les équivalents) et **Core** (c'est-à-dire le cœur ou noyau), ainsi que plusieurs autres. Les normes CEI 61968-3 à CEI 61968-9 et CEI 61968-13 exigeaient des extensions apportées au modèle CIM comme cela est spécifié par la CEI 61970-301 afin de décrire les objets et propriétés associées qui sont pertinents à la modélisation de la distribution et aux échanges d'informations applicables non seulement aux systèmes types de salles de commande, mais aussi aux systèmes d'entreprises et des partenaires, ainsi qu'aux systèmes et dispositifs de comptage. Par conséquent, de même que les applications dans le domaine de la distribution utilisent des classes issues du CIM de base, de même les applications hors du domaine de la distribution pourraient utiliser des classes définies dans le présent document (par exemple, extensions pour le marché dans la CEI 62325-301).

La Figure 2 montre les paquetages définis pour les extensions du CIM pour la distribution de la CEI 61968-11. Les notes situées sur la gauche de la figure indiquent la partie de la CEI 61968 qui commande la définition des classes dans le paquetage respectif. Remarquer toutefois que différents documents dans la série CEI 61968, ainsi que différentes applications qui utilisent le CIM pour l'échange d'informations, définiront typiquement des charges utiles de message en utilisant des classes issues de plusieurs paquetages, notamment certaines qui sont définies hors du présent document.

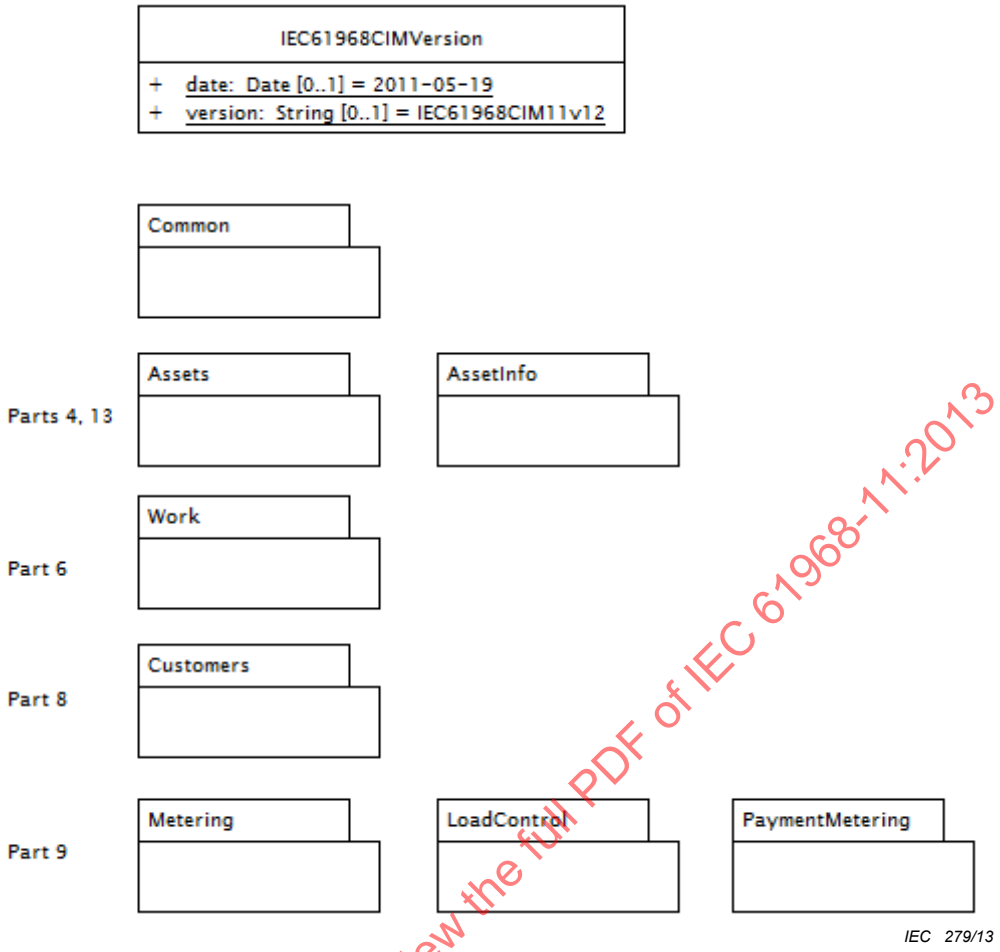


Figure 2 – Paquetages de niveau supérieur d'extensions CIM pour la distribution (DCIM)

L'Article 6 contient la spécification pour chacun des paquetages du CIM pour distribution.

NOTE Le contenu du CIM défini dans la présente spécification a été généré automatiquement à partir de la version de modèle électronique UML du CIM IEC61968CIM11v13.

4.3 Modélisation UML du CIM

4.3.1 Généralités

Le modèle CIM est défini et maintenu en utilisant le langage UML. La source et le point de maintenance pour le modèle CIM sont actuellement un fichier Enterprise Architect (EA). Celui-ci permet de visionner et de maintenir graphiquement le modèle. Le même outil est utilisé pour générer des pages web qui peuvent être visionnées sur internet. À partir du fichier EA, il est aussi généré un fichier XMI qui peut être utilisé dans les divers outils pour générer des charges utiles de message spécifiques au contexte au format XSD, RDF ou OWL pour les normes CEI 61968-3 à CEI 61968-9 et CEI 61968-13.

4.3 a pour objectif de définir un certain nombre de principes relatifs aux échanges de modèles d'informations de la CEI 61968 et d'informations associées. Pour une description des différents éléments UML utilisés dans le modèle CIM, se référer à 4.3 «Classes CIM et relations» de la CEI 61970-301:—.

4.3.2 Domaine d'application du modèle UML

La présente norme et les modèles associés ne visent pas à définir des modèles qui satisfont à toutes les exigences relatives aux informations, car cela serait une tâche impossible. Le modèle normalisé est un modèle de données canonique pour l'intégration des systèmes

d'entreprise et il est donc nécessaire qu'il satisfasse aux exigences relatives aux échanges d'informations parmi différents systèmes, c'est-à-dire, les charges utiles de message définis dans les normes CEI 61968-3 à CEI 61968-9:2009 et CEI 61968-13. Les extensions personnalisées constituent l'objet de projets et de produits non normalisés. Elles peuvent être utilisées pour s'appuyer sur le CIM pour des modèles d'informations d'applications ou systèmes spécifiques ou bien à des fins d'intégration non normalisée. En revanche, les extensions personnalisées ne sont pas maintenues par la CEI.

Le modèle DCIM global a évolué depuis de nombreuses années, mais n'a été publié comme norme CEI qu'en 2010. Pour cette première édition, le modèle UML a été divisé en classes normatives et informatives et leurs relations. Seules les classes normatives, avec leurs attributs et relations normatives sont documentées complètement dans l'Article 6. Au moment de l'édition, les classes normatives et leurs relations sont celles exigées pour la CEI 61968-4, la CEI 61968-9 et la CEI 61968-13[7]. Elles sont considérées stables et sont censées peu changer.

En revanche, les classes informatives et leurs relations informatives ne sont pas documentées dans la présente spécification. Elles sont présentes dans le modèle UML électronique et seront promues classes normatives étape par étape, avec les nouvelles éditions des normes CEI 61968-3 à CEI 61968-9 et CEI 61968-13. Certaines de ces classes seront conservées informatives tant qu'elles seront considérées instables et susceptibles de changer, et les autres pourraient être supprimées parce qu'elles ne participent pas à l'échange normalisé d'informations.

NOTE Les classes du prochain ensemble prévu être promu d'informatif à normatif dans le DCIM 12 pour la 3e édition de la présente norme CEI 61968-11 sont celles nécessaires pour appuyer le cycle de maintenance des normes CEI 61968-3, CEI 61968-4, CEI 61968-9:2009 et CEI 61968-13 et pour la 1ère édition de la CEI 61968-6 et de la CEI 61968-8.

4.3.3 Extensibilité

L'intention est pleinement de permettre des extensions au modèle d'échange d'informations. Il convient que les extensions utilisent un espace de noms local. L'espace de noms doit aussi servir à identifier l'origine de la classe ou de la propriété.

Une extension peut être définie dans un modèle UML et/ou dans un modèle physique tel que le schéma XML:

- Si une extension est faite dans le modèle UML, il convient qu'elle soit faite dans un paquetage différent comme couche ou contexte de modèle distinct(e) au lieu d'être directement ajoutée ou modifiée dans le modèle CIM. Une valeur étiquetée (tagged value) «targetNamespace» peut être utilisée pour cela si cette extension est voulue pour une génération de XSD.
- Si une extension est faite pour XSD uniquement, un espace de noms cible doit être différent d'un espace de noms XSD du CIM afin d'identifier l'origine de l'extension.

L'espace de noms de l'extension peut suivre une convention de nommage: http://<organisation>/<contrôle de version>/<profil>.

4.3.4 Définition (profil) de charge utile de message

4.3.4.1 Généralités

Un profil est un sous-ensemble des classes, associations et attributs du DCIM nécessaires pour accomplir un type spécifique d'interface. Un profil définit la charge utile de message à partir du modèle CIM (la présente norme CEI 61968-11 et la CEI 61970-301), du format d'en-tête (à partir des profils de mise en oeuvre de la CEI 61968-100 ou de la CEI 61970-552) et

7 Le seul profil défini dans le projet de la seconde édition de la CEI 61968-3 au moment de la rédaction du présent document, est totalement pris en charge par les classes du CIM de base, CEI 61970-301.

des informations à échanger (spécifiques au document de profil CEI 61968-3 à CEI 61968-9 et CEI 61968-13).

Une façon de définir les charges utiles de messages normalisés (profils) consiste à utiliser des applications telles que les divers outils libres disponibles permettant de prendre en charge la création de profil CIM. D'autres méthodes et outils peuvent être utilisés, y compris le codage manuel.

À l'heure actuelle, deux grammaires du XML sont utilisées pour la définition des profils DCIM dans la série CEI 61968.

4.3.4.2 Syntaxe du schéma XML

Les CEI 61968-3 à CEI 61968-9 définissent les profils dans la syntaxe XML conformément à la CEI 62361-100. Ces profils spécifient le format et le contenu de la partie de charge utile du message qui est échangée dans divers scenarios d'intégration, comme défini dans la CEI 61968-100. Les en-têtes de messages utilisent des combinaisons de «nouns» (noms) et de verbes. Les «nouns» renvoient en général aux concepts définis au sein du modèle DCIM (charge utile) et les «verbs» (verbes) viennent à l'appui de la technologie de transport.

Le «verb» indique en général les attributs qui sont requis. Dans le cas des verbs **create/created**, en général tous les attributs définis dans un profil sont requis, alors que dans le cas des verbs **get**, **cancel/cancelled**, **delete/deleted** et **close/closed**, seuls les identificateurs d'objet sont typiquement requis. Le verb **change/changed** exige un identificateur d'objet et les valeurs des attributs à changer.

Les recommandations concernant la définition des profils DCIM pour l'échange de données sont indiquées dans la CEI 61968-1.

Les verbs définis pour être utilisés dans les interfaces conformes à la CEI 61968, ainsi que le message enveloppe sont spécifiés dans la CEI 61968-100.

Les nouns définis pour être utilisés dans les interfaces conformes à la CEI 61968 sont spécifiés dans les CEI 61968-3 à CEI 61968-9.

4.3.4.3 Syntaxe du schéma RDF

La CEI 61970-501 définit un sous-ensemble du schéma RDF qui est utilisé pour les échanges globaux de modèles de réseau. Les informations relatives au format et à l'en-tête du profil sont spécifiées dans la CEI 61970-552[8] et sont actuellement utilisées par les profils définis dans la CEI 61968-4 et la CEI 61968-13.

4.3.4.4 Contextes d'échange de données pris en charge par le DCIM

Les profils DCIM définis dans la CEI 61968 couvrent deux contextes d'échange de données principaux parmi les systèmes d'entreprise:

- Les échanges globaux de données applicables à la configuration des systèmes. Cette sorte de processus métier est également désignée gestion de modèle, gestion de données maître ou ingénierie de données. Ce processus nécessite des charges utiles de message permettant d'établir et de maintenir les relations entre les instances des différentes entités métier communes à au moins deux systèmes d'entreprise.
- Les échanges opérationnels de données (dynamique) parmi les systèmes en exploitation. Les données échangées sur cette sorte d'interface supposent la configuration préalable des entités métier dans les différents systèmes en fonction de leur représentation DCIM commune au niveau de l'interface, mais pas nécessairement au sein du système.

8 A l'étude.

Dans le cadre du présent document, les profils de syntaxe RDFS et OWL sont principalement utilisés pour les échanges globaux de données de fichier d'importants jeux de données et les profils XSD sont utilisés pour les charges utiles de messages globaux et opérationnels.

4.4 Concepts du modèle DCIM et exemples

4.4.1 Généralités

4.4 décrit un certain nombre d'exemples de modélisation dans le domaine de la distribution et des entreprises de distribution d'électricité avec DCIM. Certains concepts sont complémentaires des exemples du CIM de base présentés dans la CEI 61970-301 (tels que la modélisation type de réseaux de distribution). D'autres concepts s'appliquent de manière générale aux entreprises de distribution d'électricité (tels que emplacements locations, documents, organisations), ou de manière plus spécifique au domaine de la distribution (par exemple, comptage et clients).

4.4.2 Concepts communs

4.4.2.1 Classes-clés dans le modèle DCIM

Le CIM de base dans la CEI 61970-301 définit principalement la fonction d'éléments de réseau électrique par le truchement de la classe **PowerSystemResource** et ses sous-classes, pour les besoins d'échange d'informations dans le contexte d'applications et de systèmes de centre de commande. Le DCIM dans le présent document CEI 61968-11 ajoute un certain nombre de classes-clés pour prendre en charge (a) la description physique de ces éléments de réseau, (b) le modèle complet pour le domaine du comptage, ainsi que (c) l'échange d'informations relatif à l'exploitation du réseau et à la planification ainsi qu'à la gestion et à la commande des compteurs dans le contexte de l'ensemble de l'entreprise de distribution d'électricité (y compris les interruptions de service, les clients et la gestion des travaux).

La Figure 3 montre les classes-clés définies dans le DCIM. Leurs relations, si elles existent, ne sont pas illustrées dans cette figure mais sont indiquées dans les paragraphes qui suivent. Il existe en fait relativement peu de relations directes entre ces classes-clés DCIM par rapport aux relations plus spécifiques qui existent entre leurs spécialisations. Le diagramme UML de la Figure 3 montre que toutes les classes-clés sont des objets identifiés, c'est-à-dire que le nom de leur superclasse, **IdentifiedObject**, est affiché dans le coin droit supérieur d'une classe.

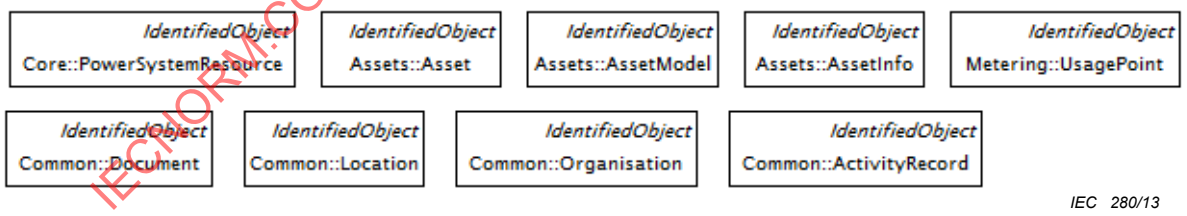


Figure 3 – Classes-clés du DCIM

Un **Asset** (c'est-à-dire un bien) est une ressource tangible de l'entreprise de services publics, y compris les équipements des systèmes électriques, les dispositifs finaux, les véhicules, les outils, les armoires, les bâtiments, etc. Pour les équipements du réseau électrique, le rôle d'un bien est modélisé par le truchement de la hiérarchie **PowerSystemResource**, définie principalement[9] dans le modèle Wires (se référer à la CEI 61970-301 et au paquetage modèle **IEC61970::Wires**). La description du bien met l'accent sur les caractéristiques physiques et la

---

9 Dans la présente partie de la CEI 61698 (seconde édition), toutes les classes DCIM définies dans le paquetage de modèle **IEC61968::WiresExt** de la CEI 61968-11:2010 (1ère édition), ont été déplacées dans le CIM de base, la CEI 61970-301 (5ème édition) et le paquetage de modèle **IEC61970::Wires**, pour assurer l'intégrité fonctionnelle du modèle global de réseau CIM, pour les réseaux de transmission et de distribution.

durée de vie de l'équipement remplissant le rôle en question et est liée à la hiérarchie **PowerSystemResource** pour la connectivité électrique.

AssetModel fournit des informations détaillées sur le bien **Asset** en tant que produit d'un fabricant donné, de sorte que un **AssetModel** et ses **AssetInfo** associées peuvent décrire plusieurs instances de **Asset**.

AssetInfo est le conteneur des informations de feuille de données sur **Asset** et **AssetModel**, et il peut également être référencé par plusieurs instances de **Asset** et **PowerSystemResource** (c'est-à-dire partagé).

Un **Location** (c'est-à-dire un emplacement) est l'endroit, le lieu ou le point de quelque chose où s'est trouvé, se trouve et/ou se trouvera quelqu'un ou quelque chose à un instant donné. Il est défini avec un ou plusieurs points de position (coordonnées) dans un système de coordonnées donné.

Un **UsagePoint** est le point logique ou physique sur le réseau auquel peuvent être affectés des relevés de comptage ou des événements de compteur. Il est utilisé à l'endroit où un compteur physique ou virtuel peut être installé. Cette classe assure le lien entre le modèle orienté réseau de l'équipement raccordé électriquement (spécialisations de **PowerSystemResource**) et le modèle orienté bien et locaux du domaine de comptage.

Un **Document** est un groupement d'informations recueillies, souvent gérées en tant que partie intégrante d'un processus métier. Il contiendra fréquemment des références à d'autres objets, tels que les biens, les personnes et les ressources du système électrique. Les **Organisations** sont les entités métier qui peuvent tenir le rôle d'entreprises de services publics, d'entrepreneurs, de fournisseurs, de fabricants, de clients, d'administration fiscale, etc.

ActivityRecord enregistre une activité pour une entité à un instant donné. L'activité peut concerner un événement déjà survenu ou une activité projetée.

Les paragraphes suivants examinent de façon plus détaillée les classes-clés DCIM, leurs spécialisations et relations et indiquent leur usage prévu.

4.4.2.2 Emplacements et représentations géographiques

Pour permettre de localiser les différentes entités et activités dans l'espace, le DCIM fournit la classe **Location**. Les entités et activités peuvent être localisées par leurs différentes sortes d'adresses ainsi que par les coordonnées de position, **PositionPoint** dans un **CoordinateSystem** (c'est-à-dire système de coordonnées) donné. Les changements de configuration d'un **Location** peuvent être suivis par les instances d'un **ActivityRecord**, **ConfigurationEvent** spécial.

La classe **Location** n'est pas destinée à être utilisée pour un échange de données graphiques, bien que ses **PositionPoint** puissent être utilisés pour échanger des coordonnées qui peuvent être utilisées pour en déduire les positions initiales pour des applications graphiques. L'échange des schémas unifilaires représentant des **PowerSystemResource**s est pris en charge par le paquetage de modèle **IEC61970::DiagramLayout** défini dans le CIM de base de la CEI 61970-301.

NOTE La prise en charge des échanges de données géospatiales complexes ne relève pas du domaine d'application de la présente CEI 61968-11, car il existe des normes largement utilisées et adoptées (par exemple, GML) pour d'autres domaines qui peuvent aussi être appliquées aux réseaux électriques.

La Figure 4 montre comment la classe **Location** peut être utilisée en relation avec des **Asset**, **PowerSystemResource** et **UsagePoint**.

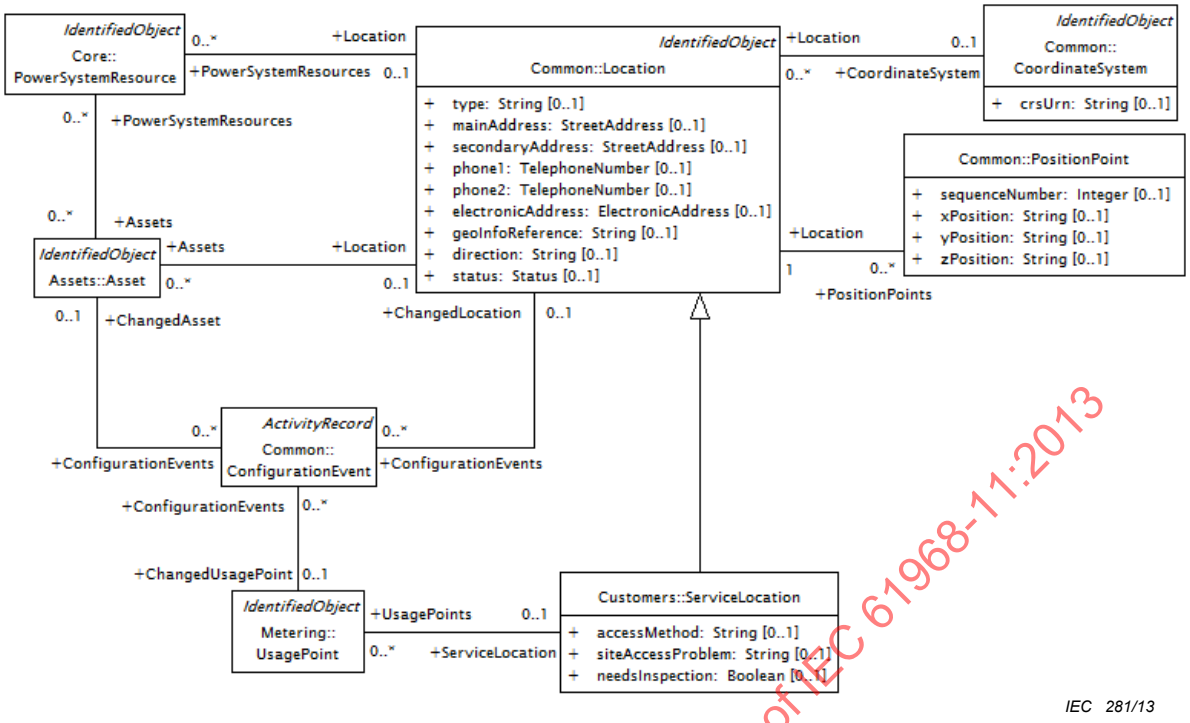


Figure 4 – Emplacements des biens et des ressources du système électrique du modèle DCIM

Un **Asset** ou un **PowerSystemResource** est directement associé à un **Location**, car le **Location** "général" fournit toutes les informations requises (telles que **mainAddress**, **electronicAddress**, ou les **PositionPoint** associés). En revanche, un **UsagePoint** fait référence à un type spécialisé d'emplacement, **ServiceLocation**, avec des attributs supplémentaires uniquement applicables au service, tels que **needsInspection**. Il s'agit d'un modèle couramment utilisé dans le DCIM: pour fournir les associations entre des spécialisations le cas échéant et associer les classes de niveau élevé lorsque la relation générique s'applique à toutes les spécialisations.

4.4.2.3 Organisations

La Figure 5 illustre les deux principales classes utilisées pour modéliser les organisations, **Organisation** et **OrganisationRole**, ainsi que leur usage prévu.

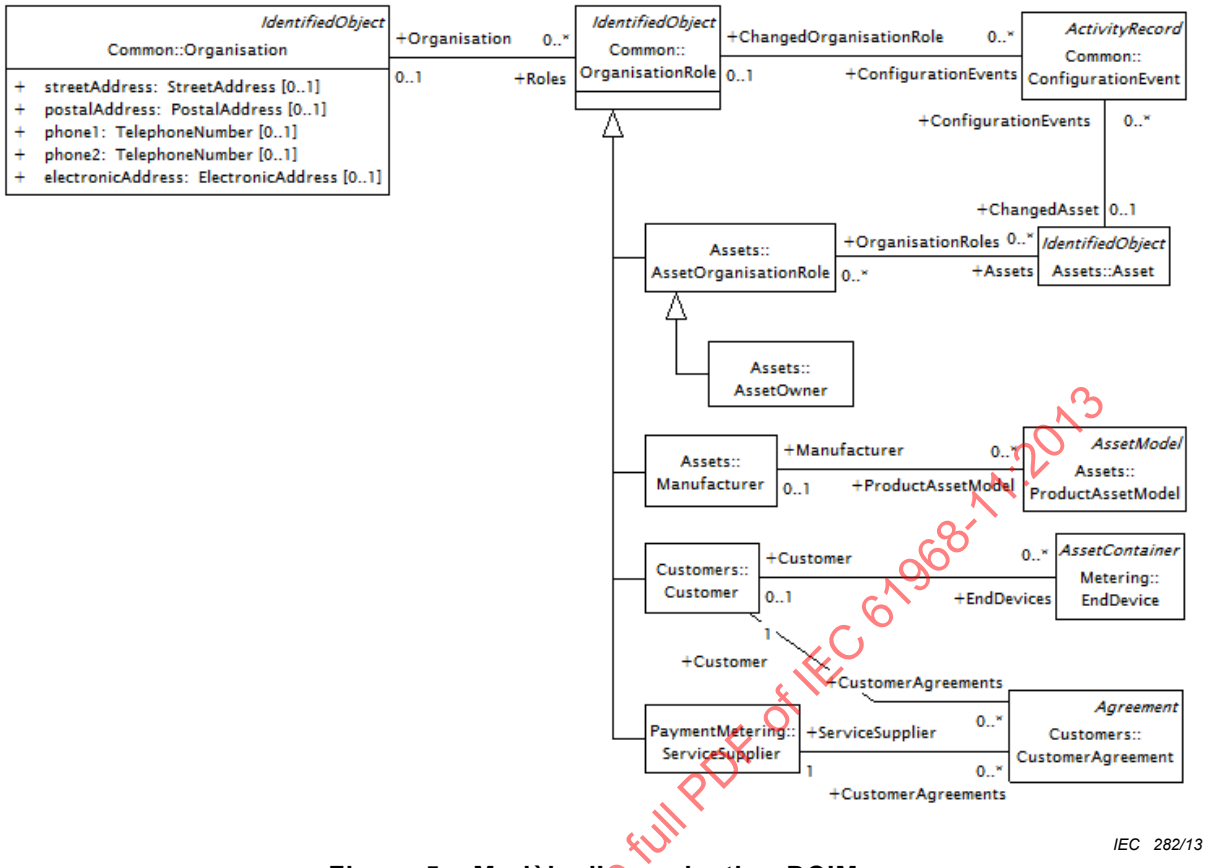


Figure 5 – Modèle d'organisation DCIM

Les entités métier modélisées avec DCIM sont souvent utilisées dans un grand nombre de contextes disparates. Par exemple, la même instance de **Organisation** peut concerner le rôle de client **Customer** dans un contexte traitant de demandes de services pour des **EndDevice**, et le rôle d'un fabricant **Manufacturer** dans un **ProductAssetModel** dans un contexte traitant de la maintenance d'un bien; ou le rôle d'un **AssetOwner** dans le contexte de gestion de bien et le rôle d'un **ServiceSupplier** dans un contexte traitant de demandes de services, conformément à **CustomerAgreement**. La modélisation est réalisée dans le DCIM avec l'association d'une **Organisation** avec plusieurs **OrganisationRole**, tandis que chaque **OrganisationRole** peut faire référence à une seule **Organisation**, en tant que entité métier.

La Figure 5 montre le **OrganisationRole** en tant que supertype commun des différents rôles. Ils sont introduits selon les besoins lorsqu'il existe des attributs spécifiques (non montrés sur la figure) et/ou des associations applicables uniquement à un domaine d'usage donné. Un exemple de ce type est **Manufacturer**: ce rôle s'applique à un **ProductAssetModel** donné, et non à **Asset** en général. Il existe par conséquent une association explicite de cette spécialisation de **OrganisationRole** spécifique au **ProductAssetModel**, tandis que d'autres rôles spécifiques sont associés aux **Asset** (par exemple, **AssetOwner**).

Dans le DCIM, **Organisation** en tant que entité métier n'est jamais spécialisée.

Les changements de configuration des **OrganisationRole** peuvent également être suivis avec les instances de **ConfigurationEvent**.

4.4.2.4 Assets

4.4.2.4.1 Classes principales dans le modèle de biens

La CEI 61968-11:2010 (1ère édition) comportait les classes **Asset** et **AssetModel** uniquement requises dans le contexte du comptage (CEI 61968-9), la présente partie de la CEI 61968

(seconde édition) fournit pour sa part un modèle de biens plus élaboré, venant à l'appui des besoins en matière d'échanges de données dans le contexte des échanges de modèle de réseau (CEI 61968-4 et CEI 61968-13). La Figure 6 donne une présentation générale des principales classes comprises dans le modèle de biens DCIM.

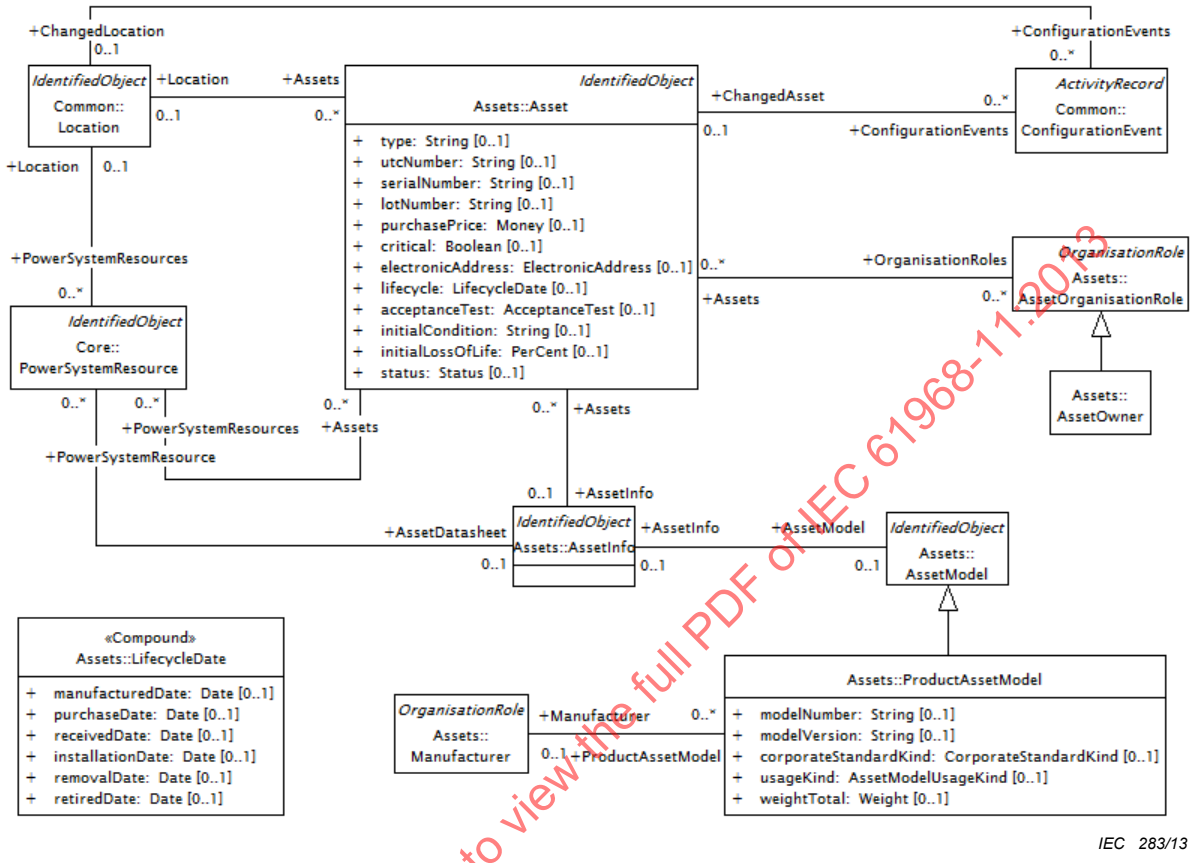


Figure 6 – Modèle de biens DCIM

Asset est une ressource tangible de l'entreprise de services publics, y compris les équipements des systèmes de puissance, les différents dispositifs finaux, les armoires, les véhicules, les bâtiments, etc. Une instance **Asset** est une ressource gérée qui peut être en fonctionnement ou en stock. Elle est caractérisée par son attribut **lifecycle**, dont les valeurs possibles sont définies par le compound **LifecycleDate**. Il est possible de la décrire avec un **Location** et sa configuration peut être suivie avec les instances de **ConfigurationEvent**. Un bien – Asset – peut être associé aux organisations au moyen de plusieurs **AssetOrganisationRole**, tels que **AssetOwner**.

NOTE La présente partie de la CEI 61968 (seconde édition) (documentant le modèle DCIM11) comporte une seule spécialisation de **AssetOrganisationRole**. Il est prévu que la prochaine édition (documentant le modèle DCIM12) ajoutera plus de rôles d'organisation pour ce qui concerne les biens, tels qu'agents de maintenance ou utilisateur, afin de prendre en charge les échanges de données exigées par la CEI 61968-3 et la CEI 61968-6.

Certains échanges de données nécessitent le modèle de connectivité des équipements électriques, auquel cas un **Asset** peut faire référence à une ou plusieurs instances **PowerSystemResource**, qui modélisent la fonction électrique de l'équipement représentée par le **Asset** physique – voir également 4.4.2.5. L'inverse s'applique également, à savoir que certains échanges de données entre les systèmes opérationnels et d'autres systèmes (tels que les systèmes de gestion de planification ou de biens) nécessitent une vue physique associée au modèle électrique du système électrique, auquel cas un **PowerSystemResource** peut faire référence au **Asset** applicable et/ou aux instances **AssetInfo**. En dernier lieu, certains échanges de données ignorent totalement le modèle de connectivité électrique et fonctionnent exclusivement avec le modèle de biens.

Une autre classe-clé du modèle de biens est **AssetInfo**, qui est un conteneur identifiable pour différents paramètres physiques de description des **Asset** associés. Un **Asset** sans l'instance concrète associée **AssetInfo** n'est pas totalement décrit. **AssetInfo**, qui modélise ce qui est parfois appelé “catalogue”, “bibliothèque” ou “code”, est un jeu d'attributs d'un bien (asset), représentant les informations de feuille de données types d'un dispositif physique qui peut être instancié et partagé dans différents contextes d'échange de données:

- comme attributs d'une instance asset (installé ou en stock);
- comme attributs d'un modèle de biens (produit d'un fabricant);
- comme attributs d'un asset type (type générique d'un asset utilisé pour les conceptions/planification des extensions)[10].

Dans la mesure où les attributs de feuille de données des dispositifs physiques dépendent du type de dispositif qu'ils décrivent, **AssetInfo** est la classe spécialisée avec les types concrets qui détiennent les attributs et les associations spécifiques à la spécialisation concernée (voir 4.4.2.4.2).

La principale caractéristique d'une instance concrète **AssetInfo** réside dans le fait qu'il est possible d'y faire référence à partir de plusieurs instances de **PowerSystemResource** ou **Asset**. Il s'agit d'une exigence majeure applicable à la modélisation de réseau de distribution lorsque les caractéristiques de l'équipement sont rarement définies par **PowerSystemResource** ou **Asset**, du fait de la nature et du nombre d'équipements utilisés. En règle générale, une instance d'un **AssetInfo** spécifique peut être référencée par des milliers d'instances de **PowerSystemResource** ou **Asset**. Une exigence similaire s'applique aux paramètres électriques calculés, tels que les impédances de ligne ou de transformateur, ce qui est reflété par les classes de catalogue électrique applicables (voir 4.4.3.3.1).

La troisième abstraction majeure associée aux biens (assets) est le concept de **AssetModel**. Il représente le modèle d'un **Asset**, soit le produit d'un fabricant particulier (**ProductAssetModel**) soit un modèle de biens générique ou élément de matériau (non montré sur la Figure 6, car cette classe est encore informative). Les caractéristiques de feuille de données du **AssetModel** sont disponibles par le biais de la spécialisationde **AssetInfo** associée et peuvent être partagées avec les instances **Asset** ou **PowerSystemResource**.

4.4.2.4.2 Spécialisations du modèle de biens

Les premières versions du modèle UML électronique du DCIM utilisaient trois hiérarchies d'héritage parallèles associées aux biens, en complément à la hiérarchie d'héritage **PowerSystemResource** du CIM de base de la CEI 61970-301. La présente partie de la CEI 61968 (seconde édition) présente un modèle de biens simplifié, bien consolidé avec le modèle de fonction de **PowerSystemResource**, et avec les classes majeures introduites en 4.4.2.4.1.

4.4.2.4.2 présente les lignes directrices de modélisation établies et appliquées dans la présente édition de la CEI 61968-11, et qu'il convient de suivre dans les futures éditions normatives du DCIM ainsi que pour les extensions personnalisées du DCIM normalisé.

a) Parmi les classes **Asset**, **AssetModel**, **ProductAssetModel** et **AssetInfo**: il convient que **ProductAssetModel** ne soit jamais spécialisé dans le DCIM normalisé, et **AssetModel** n'est spécialisé qu'avec **ProductAssetModel**[11]. **AssetInfo** est toujours spécialisé. **Asset** peut être spécialisé selon la ligne directrice d) uniquement.

---

[10] Parmi les classes informatives du modèle UML électronique du DCIM, il existe une autre spécialisation de **AssetModel** qui est prévue pour devenir normative dans certaines des éditions ultérieures de la CEI 61968-11.

[11] Parmi les classes informatives du modèle UML électronique du DCIM, il existe une autre spécialisation de **AssetModel** qui est prévue pour devenir normative dans certaines des éditions ultérieures de la CEI 61968-11.

b) Si un dispositif/équipement existe dans le paquetage de modèle **IEC61970::Wires** de la CEI 61970-301 (comme spécialisation de, ou associé à **PowerSystemResource**), il n'est pas admis de définir ses homologues comme une spécialisation de **Asset**. En revanche, une spécialisation de **AssetInfo** doit être définie avec ses propriétés de feuille de données, le cas échéant.

c) La relation entre les modèles fonctionnels et physiques est établie entre les spécialisations applicables de **PowerSystemResource** et **AssetInfo** au moyen de la relation héritée, soit **PowerSystemResource–AssetInfo** (si l'instance asset n'existe pas) ou **PowerSystemResource–Asset–AssetInfo** (si l'instance asset est disponible).[12]

d) Si une entité de dispositif/équipement/réseau n'est pas modélisée dans le paquetage de modèle **IEC61970::Wires** de la CEI 61970-301, il convient qu'elle hérite de **Asset** ou **AssetContainer**, et il n'est pas nécessaire que son nom comporte le terme (désormais redondant) “Asset”.[13]

e) Selon a), il convient d'éliminer les spécialisations de **Asset**, **AssetModel**, **ProductAssetModel** comme suit: il convient de déplacer les attributs et extrémités d'associations nécessaires soit dans une spécialisation d'**Asset** (s'ils s'appliquent à une instance de **Asset**), ou dans une spécialisation d'**AssetInfo** (s'ils représentent des informations de feuille de données, applicables à plusieurs instances **Asset**).[14]

La Figure 7 illustre des exemples de DCIM qui reflètent les lignes directrices susmentionnées et également l'usage d'autres classes issues du modèle de biens.

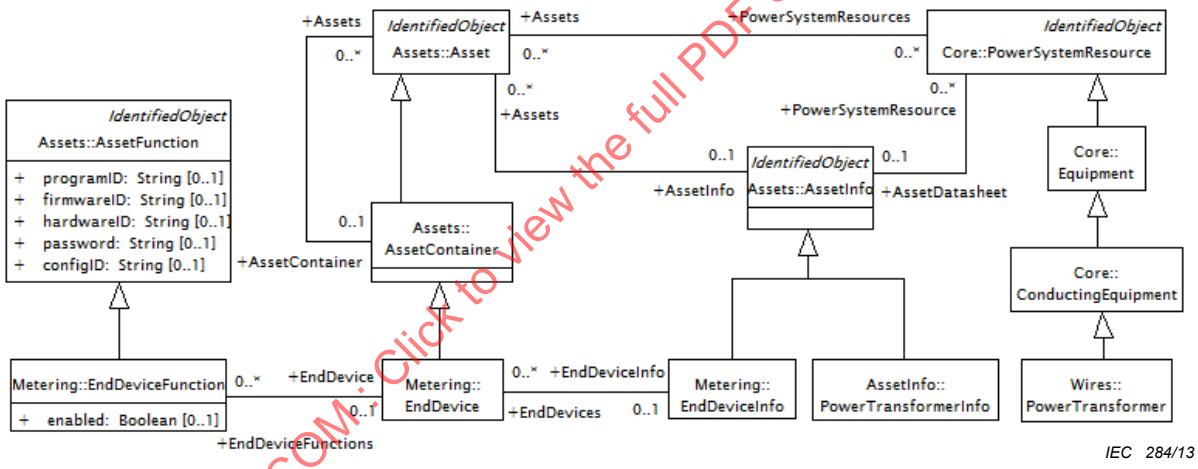


Figure 7 – Biens DCIM – lignes directrices de spécialisation

AssetContainer est la spécialisation de **Asset** et constitue également le conteneur pour plusieurs **Asset**. En tant que spécialisation, il hérite des attributs et des associations avec les autres classes. En tant que conteneur, il permet de modéliser les **Asset** composites. Un exemple de ce type est le **EndDevice**, qui peut contenir d'autres entités susceptibles d'avoir une durée de vie indépendante (telles que les modules de communication, non montrés sur la figure). D'autres exemples de **AssetContainer** seraient les poteaux ou les voûtes (actuellement non présents dans le DCIM normatif). Aucun de ces objets du monde réel n'est défini dans le modèle fonctionnel **PowerSystemResource** (ligne directrice d)).

---

12 Du fait de l'application de cette ligne directrice, certaines associations directes entre les spécialisations de **PowerSystemResource** et **AssetInfo** ont été éliminées pour pouvoir utiliser de préférence la nouvelle association héritée définie entre **PowerSystemResource** et **AssetInfo**.

13 Du fait de l'application de cette ligne directrice, certaines classes issues du paquetage de modèle **Metering** ont été renommées pour ne pas utiliser le suffixe “Asset”.

14 Du fait de l'application de cette ligne directrice, certaines classes issues du paquetage de modèle **Metering** ont “perdu” leurs attributs d'instance qui ont été déplacés vers la spécialisation **AssetInfo** correspondante.

La Figure 7 ne montre que deux des nombreuses spécialisations de **AssetInfo**: **EndDeviceInfo** et **PowerTransformerInfo**. La dernière hérite d'une association aux **PowerTransformer** soit par **AssetInfo–PowerSystemResource** ou **AssetInfo–Asset–PowerSystemResource** (ligne directrice c); pour de plus amples informations, voir 4.4.3.4). Il n'existe aucune spécialisation d'**Asset** correspondant à **PowerTransformer** (ligne directrice b)).

S'agissant de **EndDeviceInfo**, il est directement associé à plusieurs instances de **EndDevice**, et il hérite également de l'association **AssetInfo–Asset**. Dans ce cas, il s'agit de la relation la plus spécifique qu'il convient d'utiliser. A l'heure actuelle, il existe peu de spécialisations de **Asset**, et il est acceptable de définir ces types d'associations concrètes entre spécialisations, même si l'association héritée existe. Pour ce qui concerne la relation entre les spécialisations d'**AssetInfo** et de **PowerSystemResource**, elles ont été considérées comme trop nombreuses pour pouvoir définir toutes les relations explicites qui les relient.

En dernier lieu, la classe **AssetFunction** et sa spécialisation **EndDeviceFunction** démontrent la conception préférée, lorsqu'elle est réalisable: commencer uniquement avec les associations directes entre les spécialisations applicables, telles que **EndDevice–EndDeviceFunction** de la Figure 7. Dans les futures éditions du DCIM, si plusieurs cas nécessitant une relation entre les spécialisations de **AssetFunction** et de **Asset** ou de **AssetContainer** sont identifiés, la relation concrète serait vraisemblablement remplacée par une relation générique.

4.4.2.5 Modèle électrique contre modèle physique

Le CIM de distribution couvre la représentation tant électrique que physique d'un objet. La classe **PowerSystemResource** modélise la représentation électrique et est souvent utilisée pour l'exploitation du réseau, la surveillance, la gestion des interruptions de service et la planification des opérations, alors que la classe **Asset** modélise la représentation physique d'un objet et est surtout utilisée pour la gestion de biens et de travaux, mais également pour la planification des extensions de réseau, la gestion des interruptions de service et les études de réseau. Par exemple, la représentation physique peut être essentielle dans la dérivation des attributs pour la représentation électrique, même si aucune classe explicite **Asset** n'est utilisée. Par exemple, 4.4.3.3.3 explique comment le modèle de biens peut aider à calculer la caractéristique électrique d'une ligne de distribution. La relation entre les deux aspects de l'équipement électrique fournit aussi des informations cruciales pour la planification des extensions, la gestion des travaux et la gestion des interruptions de service.

La relation entre **Asset** et **PowerSystemResource** permet la navigation entre les deux représentations (modèles) différentes d'un objet du monde réel. Les aspects de connectivité et opérationnels (tels que limites opérationnelles ou statut d'alimentation en énergie) sont toujours modélisés par le truchement de **PowerSystemResource** et de ses spécialisations, alors que les aspects physiques (tels que caractéristiques nominales dans les feuilles de données, dimensions, cycle de vie des biens et données financières) sont toujours modélisés par le truchement de la classe **Asset** et de ses classes connexes.

Il est à noter que **Asset** et **AssetInfo** ont des relations avec **PowerSystemResource**. La relation entre **PowerSystemResource** et **Asset** est généralement utilisée pour les échanges de données lorsque le **Asset** est en fonctionnement afin de donner le point de vue physique de l'équipement de réseau électrique en étant exploité, ou dans l'autre direction pour donner le point de vue de la connectivité électrique au système fonctionnant avec la représentation physique. La relation entre **PowerSystemResource** et **AssetInfo** peut être utilisée même s'il n'existe aucune instance réelle d'**Asset** disponible ou nécessaire. Des exemples types sont les applications de planification ou les applications de calcul de réseau utilisées dans les systèmes opérationnels (tels que les calculs de répartion (power flow) de distribution).

4.4.3 Concepts et exemples de modélisation de réseau

4.4.3.1 Connectivité et modélisation de phase

La connectivité du modèle de distribution suit le modèle de connectivité CIM, c'est-à-dire qu'elle s'appuie sur les classes **ConductingEquipment**, **Terminal** et **ConnectivityNode** (se référer à la CEI 61970-301:—, 4.4.4, Modèle de connectivité et 4.4.8, Modélisation de fils de phase). L'attribut **Terminal.phases** permet d'exprimer les informations de phase de chaque **ConductingEquipment** pour un réseau de distribution polyphasé. Il permet également de représenter les cas de désadaptation normale de phase de réseau. Dans les réseaux de distribution, il existe quelques rares cas où un organe de coupure ouvert normal peut être connecté à la phase A d'un côté et à la phase B de l'autre côté. Les **Jumper** peuvent introduire une désadaptation normale de phase tant qu'un côté est hors tension avant de connecter le **Jumper**. Par exemple, un **Jumper** peut relier une phase A normale sous tension à une phase B normale effectivement hors tension. Ce cas n'est admis que lorsque toutes les charges en aval sont sur la ou les mêmes phases.

Pour un exemple élaboré du réseau exemplaire triphasé équilibré, se référer à la CEI 61970-301:—, 4.4.4.2, Exemple de connexité et d'emboîtement, et pour un exemple illustrant la manière dont il est possible d'utiliser le modèle de connectivité CIM pour représenter un réseau polyphasé, se référer à la CEI 61970-301:—, 4.4.8, Modélisation de fils de phase.

La Figure 8 donne une présentation générale des classes de phase généralement utilisées pour la modélisation et les applications de réseau de distribution.

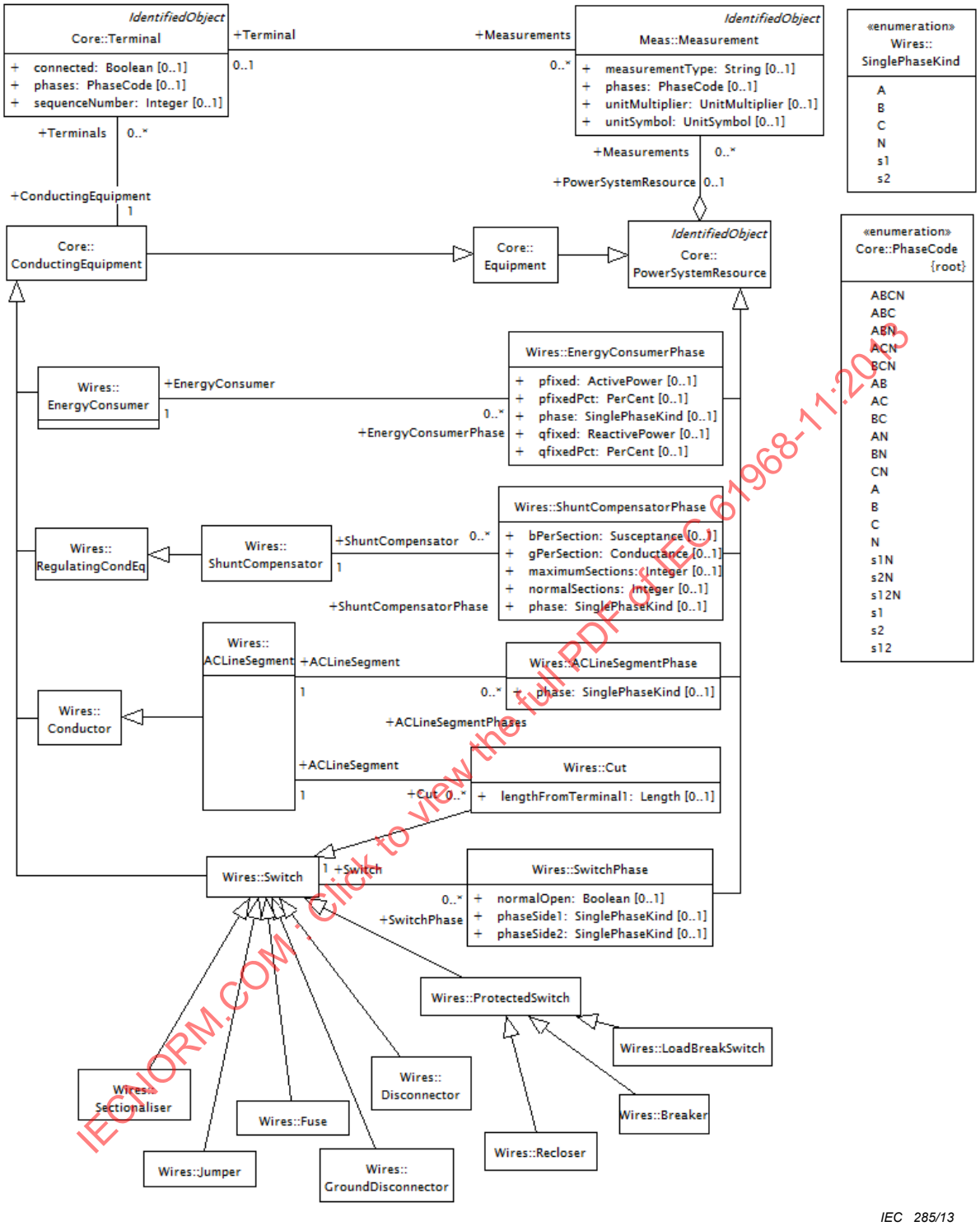


Figure 8 – Modélisation de phase DCIM

Dans les réseaux de distribution, certains dispositifs peuvent présenter des caractéristiques électriques et opérationnelles différentes sur chaque phase. Par exemple, un **Switch** (c'est-à-dire organe de coupure) triphasé (qu'il s'agisse d'un **Breaker**, un **Fuse** ou toute autre spécialisation de **Switch**) peut être constitué de trois organes de coupure physiques séparés, un pour chaque phase. Dans la majorité des cas, les trois organes de coupure individuels ont les mêmes propriétés physiques. S cet organe de coupure est à commande simultanée (c'est-à-dire que tous les organes de coupure physiques ont le même état de fonctionnement normal et/ou en temps réel), il peut être modélisé avec une seule instance de **Switch**. Cependant, il est possible que les phases individuelles aient différentes propriétés physiques. Par exemple,

dans le cas d'un **Fuse**, d'un point de vue au moins théorique, il est possible d'utiliser différentes caractéristiques de fusible sur chaque phase. De même, s'agissant de organe de coupures non à commande simultanée, il est possible que leur état de fonctionnement normal et/ou en temps réel soit différent pour chaque phase. Dans ce cas, la modélisation est réalisée avec une instance de **Switch** contenant trois instances de **SwitchPhase**, une pour chaque phase. Il est important de noter que dans chaque cas il existe une instance de **Switch**, et que cette instance peut être référencée à partir de différents contextes (par exemple, configuration, opérations).

NOTE La présente édition de la CEI 61968-11 ne prend pas en charge les organes de coupure à plusieurs états (modélisés par **CompositeSwitch**).

Dans le cadre de cette deuxième édition de la CEI 61968-11, la classe **Measurement**, de manière analogue à la classe **Terminal**, a défini l'attribut **phases** afin de fournir une description de phase détaillée applicable à une mesure. Dans la plupart des cas, l'attribut est facultatif dans la mesure où la mesure réalisée sur une borne **Terminal** est adaptée à sa phase. Cependant, pour un **ConductingEquipment** qui peut fonctionner de manière séparée sur chaque phase individuelle, l'attribut **Measurement.phases** donne la capacité de spécifier des mesures sur chaque phase. Sur la base du modèle de commande CIM défini dans le CIM de base, CEI 61970-301:—, 4.4.10 Mesures et commandes, les commandes peuvent également être appliquées aux phases séparées.

Dans le cadre de la présente partie de la CEI 61968, le modèle de distribution CIM permet de prendre en charge la représentation des changements temporaires qui se produisent dans le réseau. Pour une description détaillée et un exemple, se référer à la CEI 61970-301:—, 4.4.9 Modèle de coupures, colliers et cavaliers.

4.4.3.2 Charges monophasées et déséquilibrées

La Figure 9 montre les classes disponibles pour modéliser les charges de distribution, qui sont souvent déséquilibrées parmi les trois phases à un emplacement. Dans certains cas, il se produit des charges monophasées et diphasées.

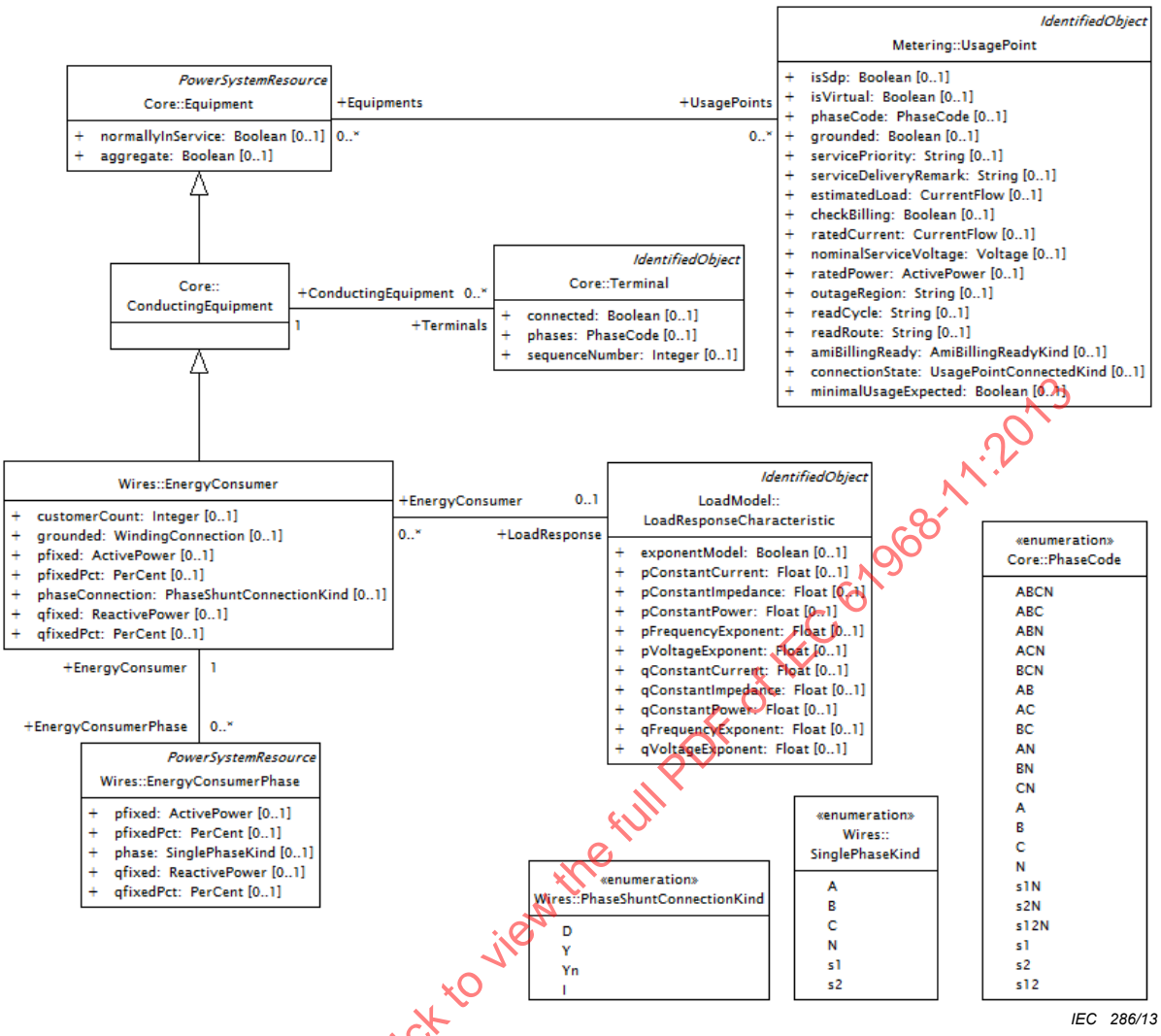


Figure 9 – Modèle de charge du DCIM

Il convient d'utiliser la classe **EnergyConsumer** pour instancier le modèle de charge, qui peut facultativement être associé à un compteur par le truchement de **UsagePoint**. **EnergyConsumer** hérite de **ConductingEquipment** qui est associé à **Terminal**. **Terminal** a l'attribut **phases** auquel peut être assigné un libellé d'énumération **PhaseCode**. Par exemple, **AN** décrit une charge monophasée de A au neutre, **BC** décrit une charge monophasée de B à C, **ABCN** décrit une charge triphasée étoile à la masse, **ABC** décrit une charge triphasée à connexion delta, etc.

Il convient que l'attribut **phaseConnection** de **EnergyConsumer** soit **Y** ou **Yn** pour une charge à connexion en étoile ou phase-neutre; il convient qu'il soit **D** pour une charge à connexion delta ou entre phases. Pour une charge triphasée équilibrée, spécifier la puissance totale réelle et réactive sur les attributs **EnergyConsumer**, pour obtenir une distribution uniforme sur les trois phases.

Si une charge triphasée est déséquilibrée, elle peut être modélisée par la même instance **EnergyConsumer**, faisant désormais référence aux trois instances supplémentaires **EnergyConsumerPhase**, chacune représentant une phase avec son attribut **phase**. Cela permet également d'utiliser un **EnergyConsumer** avec son identité pour la modélisation de charges déséquilibrées. Avec les charges monophasées ou diphasées, il convient que le modèle inclue également des conducteurs ou des transformateurs en phase appropriée qui établissent la connectivité de retour à la source. Il n'est pas nécessaire de spécifier la puissance totale réelle et réactive sur les attributs **EnergyConsumer**, dans la mesure où ils

peuvent être dérivés de la distribution entre phases spécifiée sur les attributs **EnergyConsumerPhase**. Chaque **phase** peut être **A**, **B**, **C**, **S1** ou **S2** (**N** ne s'applique pas à ce cas). Pour une charge à connexion delta ou entre phases, indiquée par **D** pour **phaseConnection**:

- utiliser **A** pour la charge connectée entre les phases **A** et **B**;
- utiliser **B** pour la charge connectée entre les phases **B** et **C**;
- utiliser **C** pour la charge connectée entre les phases **C** et **A**;
- utiliser **S1** pour la charge connectée entre les phases **S1** et **S2**.

Lorsque la charge est une combinaison de courant constant, de puissance constante ou d'impédance constante, il convient d'associer une instance de **LoadResponseCharacteristic** à l'instance de **EnergyConsumer**.

NOTE La présente partie de la CEI 61968 ne fournit aucun moyen permettant de définir différentes caractéristiques de charge par phase.

4.4.3.3 Segments de ligne de distribution

La Figure 10 montre les classes disponibles pour modéliser les segments de lignede CA (c'est-à-dire, des conducteurs). Il y a trois façons de décrire les paramètres d'impédance d'un **ACLineSegment**, en utilisant uniquement le paquetage **Wires**, et une quatrième façon utilisant le paquetage **AssetInfo**.

NOTE La présente partie de la CEI 61968 (documentant le modèle DCIM11) et l'édition correspondante de la CEI 61970-301 (document le modèle CIM15 de base) reflètent des modèles consolidés de segment de ligne pour la transmission et la distribution (T&D). Par conséquent, les classes issues de la CEI 61968-11:2010 (1ère édition) ont été légèrement modifiées et déplacées du paquetage de modèle **IEC61968::WiresExt** au paquetage de modèle **IEC61970::Wires** du CIM de base, CEI 61970-301 (5ème édition).

Afin de représenter une ligne monophasée, diphasée ou triphasée déséquilibrée utilisant différents fils, il convient qu'une instance **ACLineSegmentPhase** soit associée à chaque phase ou un fil de neutre retenu dans le modèle. Les valeurs de l'attribut **phase** applicable sont **A**, **B**, **C** pour les lignes triphasées, **S1** et **S2** pour les circuits du secondaire monophasés, et **N** lorsque le fil de neutre est modélisé. Chaque **ACLineSegmentPhase** dispose d'une association facultative à **WireInfo**, détaillée ultérieurement, mais toutes les autres caractéristiques proviennent de son **ACLineSegment** associé.

S'agissant d'une ligne déséquilibrée, les impédances calculées reflètent les positions des fils physiques sur le pylône ou le poteau, ces positions étant généralement ordonnées de gauche à droite et du haut vers le bas en fonction de leurs coordonnées horizontales et de hauteur. Soit par exemple un poteau à console ayant 3 positions de fils numérotées 1 (plus à gauche), 2 (décalage par rapport au centre) et 3 (plus à droite), les impédances peuvent être calculées avec le fil de phase A en position 1, le fil de phase B en position 2 et le fil de phase C en position 3. Un **ACLineSegment** différent peut utiliser le même type de poteau et de fils, mais avec le fil de phase C en position 1, le fil de phase B en position 2 et le fil de phase A en position 3. Les paramètres des impédances calculées sont différents de ceux du premier cas. Ils sont mathématiquement liés aux opérations des rangées et colonnes de la matrice d'impédance, ces opérations n'étant toutefois pas prises en charge dans le CIM et il n'existe aucun concept d'ordre des phases dans le CIM. Cela implique que chaque ordre de phase physique différent nécessite une description d'impédance différente dans le CIM. Cela affecte l'utilisation de **PerLengthPhaseImpedance** et de **WireSpacingInfo** de la Figure 10.

Les lignes transposées peuvent être modélisées pour chaque section avec un **ACLineSegment** différent et une description d'impédance différente comme susmentionné. Il convient que la description d'impédance utilise **PerLengthPhaseImpedance** ou **WireSpacingInfo**, car les paramètres de séquence supposent une transposition continue. Il n'est pas nécessaire d'utiliser les instances **ACLineSegmentPhase** pour les sections de lignes triphasées transposées, si tous les types de fils de phase sont identiques dans une section.

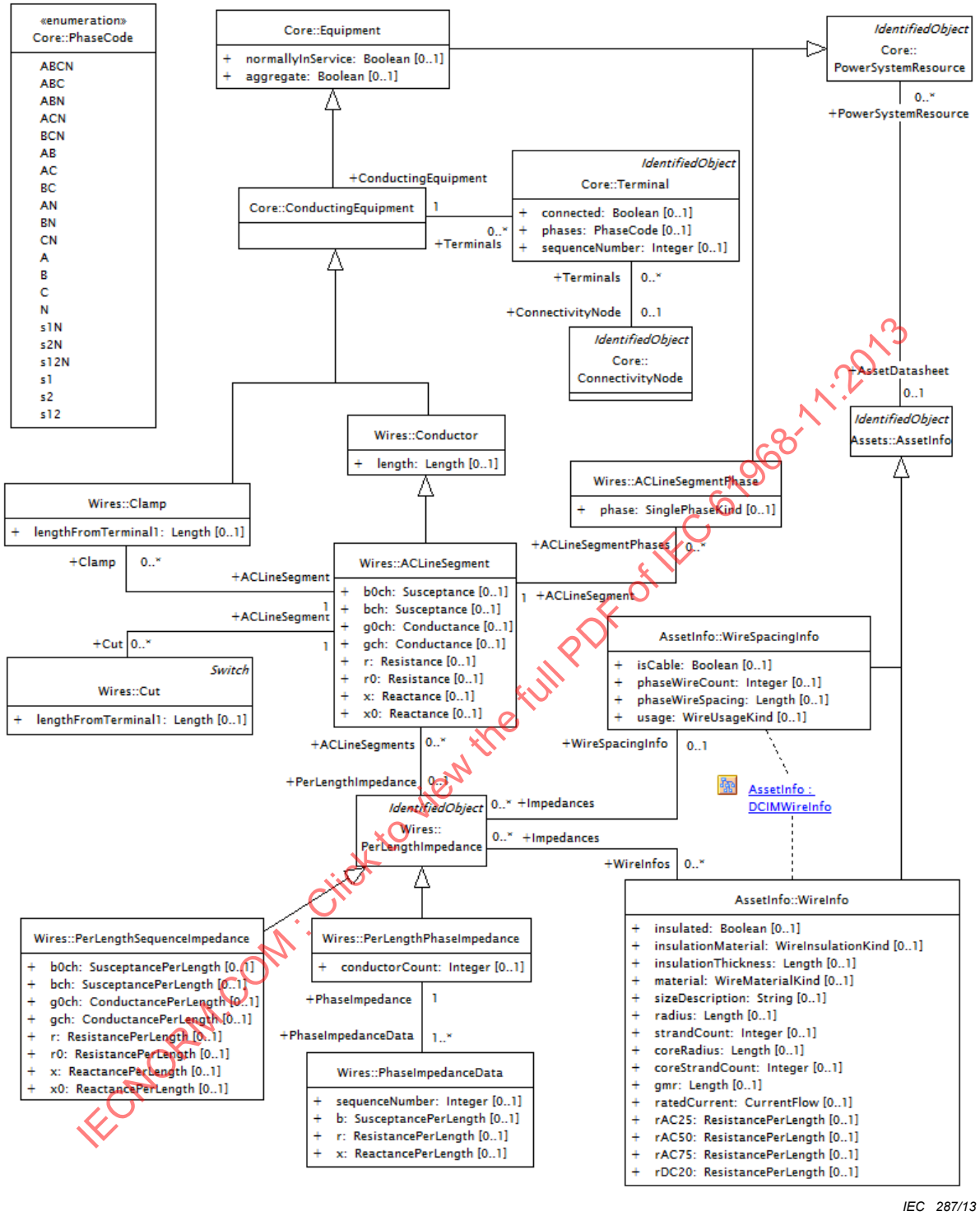


Figure 10 – Modèle de connectivité de lignes DCIM

Il existe trois manières de spécifier les impédances de ligne en utilisant uniquement le paquetage de modèle **IEC61970::Wires** de la CEI 61970-301:

- Les attributs **r**, **r0**, **x**, **x0**, **bch**, **bch0**, **gch** et **gch0** de **ACLineSegment** peuvent être utilisés pour un modèle équilibré. Voir 4.4.3.3.1 pour l'usage avec des segments de ligne monophasée et diphasée.
- Pourvoir une association à **PerLengthSequenceImpedance**. Cette classe modélise un catalogue électrique ou une bibliothèque de «codes de ligne» qui ont des impédances en séquence et un chargement de ligne linéique. Plusieurs **ACLineSegment** partagent une

instance de **PerLengthSequenceImpedance** et calculent l'impédance avec leur attribut **length** hérité de **Conductor**. Voir 4.4.3.3.1 pour l'usage des paramètres séquentiels avec des segments de ligne monophasée et diphasée.

- pourvoir une association à **PerLengthPhaseImpedance**, qui fait référence aux matrices symétriques d'impédance et d'admittance NxN pré-calculées par unité de longueur. Cette classe modélise également un catalogue électrique et nécessite l'attribut **length** hérité. Le **conductorCount** doit être au moins égal au nombre de phases, mais il pourrait être supérieur si le neutre (ou autre conducteur mis à la terre) est retenu dans la matrice. **PhaseImpedanceData** met en œuvre des éléments de matrice Z et Y, mémorisés dans l'ordre des colonnes. Les attributs **r**, **x** et **sequenceNumber** sont tous requis, alors que **b** est facultatif. Seuls les éléments triangulaires inférieurs sont mémorisés et, de ce fait, une matrice 3x3 comporterait 6 éléments (c'est-à-dire instances de **PhaseImpedanceData**). Les lignes et les colonnes de la matrice doivent être dans l'ordre des phases. Un **ACLineSegment** de référencement affecte la ligne 1 à la première phase présente issue de la liste ordonnée (**A**, **B**, **C**, **s1**, **s2**, **N**), la ligne 2 à la phase suivante présente, et ainsi de suite. Voir aussi 4.4.3.3.2 ci-dessous.

Afin de spécifier les impédances en utilisant le paquetage **AssetInfo**, l'attribut **Conductor.length** (c'est-à-dire la longueur de conducteur) est requis car les paramètres électriques de l'instance sont définis comme étant (paramètres linéiques) × (**Conductor.length**). Voir également 4.4.3.3.3 ci-dessous.

Un **WireSpacingInfo** peut être associé au **ACLineSegment** en appliquant l'une des deux méthodes suivantes:

- (**PowerSystemResource–AssetInfo**) Utiliser l'extrémité d'association héritée **AssetDatasheet** de **ACLineSegment** pour référencer **WireSpacingInfo**. Cette méthode peut également être appliquée pour associer **WireInfo** pour son attribut **ratedCurrent**.
- (**PowerSystemResource–Asset–AssetInfo**) Instancier la classe **Asset**, dans laquelle l'extrémité d'association **AssetInfo** référence le **WireSpacingInfo**, et plusieurs **PowerSystemResource** référencent chaque **ACLineSegment** en utilisant le **WireSpacingInfo** considéré.

Pour les calculs d'impédance, il est également nécessaire d'associer les **WireInfo** à chaque **ACLineSegment** ou **ACLineSegmentPhase** de référencement. Cela peut être réalisé avec le modèle de navigation de classe **AssetDatasheet** (**PowerSystemResource–AssetInfo**) ou **Asset** (**PowerSystemResource–Asset–AssetInfo**). Dès lors que le **ACLineSegment** dispose des **ACLineSegmentPhases** associées, il convient que chaque **ACLineSegmentPhase** ait son propre **WireInfo**. Dans le cas contraire:

- Au moins un **WireInfo** doit être associé au **ACLineSegment**, et il est supposé s'appliquer à tous les fils de phase. Il s'applique également à tout fil de neutre, à moins qu'il soit possible d'associer un deuxième **WireInfo** identifiable.
- Un deuxième **WireInfo** facultatif peut être associé pour le fil de neutre, identifié comme étant celui ayant la plus petite valeur de l'attribut **WireInfo.radius**. Lorsque cette hypothèse ne s'applique pas à la ligne, utiliser la modélisation de phase avec **ACLineSegmentPhase**.

La Figure 10 montre les associations entre **PerLengthImpedance**, **WireSpacingInfo** et **WireInfo**. Elles concernent cependant la gestion de bibliothèque de biens et non les calculs d'impédance. Pour les calculs d'impédance, le modèle de classe **AssetDatasheet** ou **Asset** est requis pour utiliser **WireSpacingInfo** et **WireInfo**.

La Figure 11 montre la hiérarchie de la classe **WireInfo**, utilisée pour définir les impédances de segment de ligne à partir de données physiques (parfois appelées «codes de ligne» et « codes de câble »). **WireInfo** est une classe de base qu'il convient de ne pas instancier directement. Ses attributs décrivent les données physiques d'un fil ou câble aérien (à noter que **OverheadWireInfo** dérivé n'a réellement pas d'attributs propres) ou d'un conducteur de phase avec âme de câble.

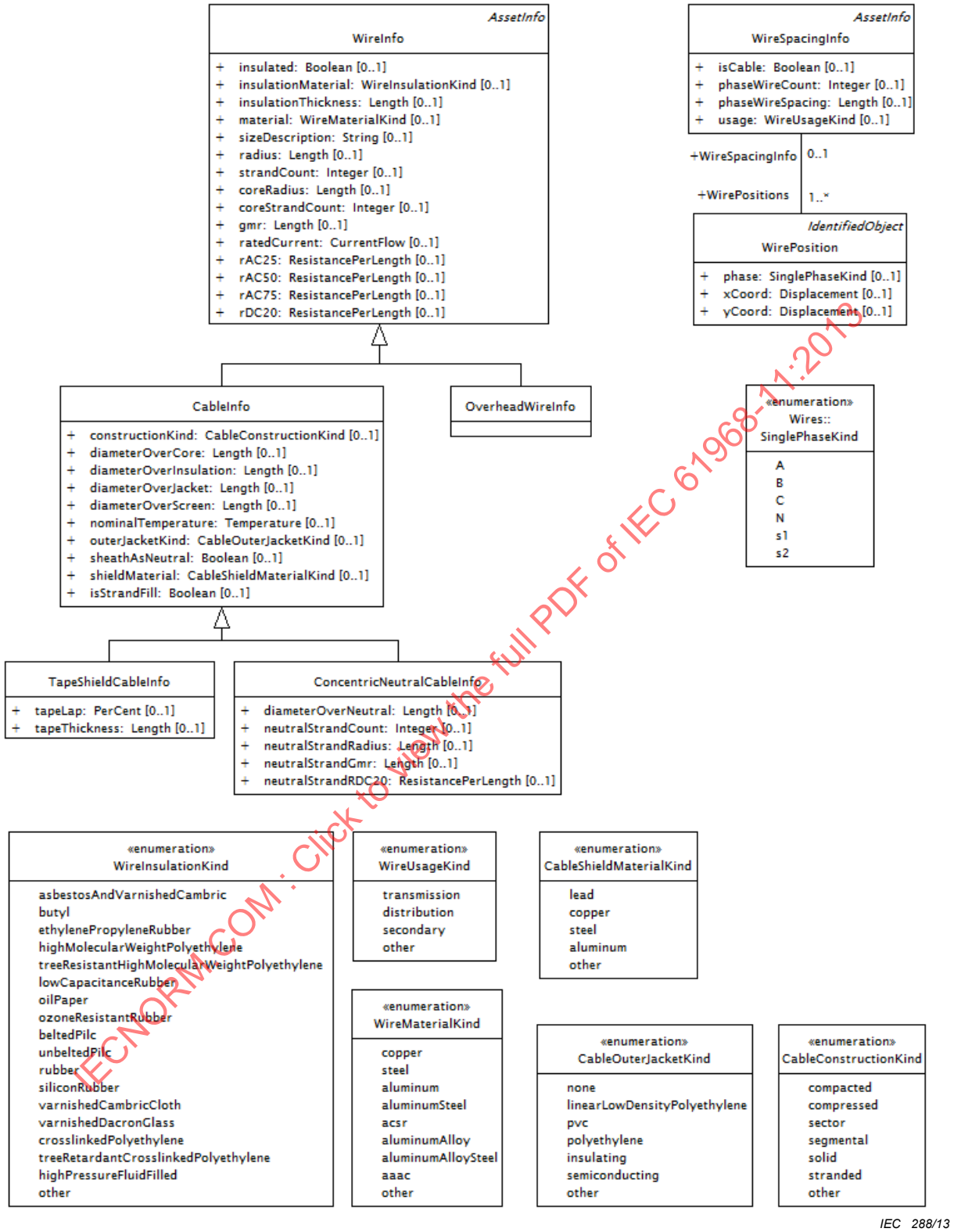


Figure 11 – Modèle (feuille de données sur les lignes et les câbles) de conducteurs DCIM

Les attributs **material**, **sizeDescription**, **strandCount** et **coreStrandCount** (pour l'ACSR, à savoir "aluminium conductor steel reinforced wire", c'est-à-dire fil conducteur en aluminium renforcé d'acier) de **WireInfo** aident à identifier le câble et sont souvent codés dans le nom local d'instance. Les attributs électriques requis au minimum sont **gmr**, **radius** et **rAC50.** Un tableau de fils complet comporterait en général des résistances définies à d'autres températures et fréquences, telles que **rAC25**, **rAC75** et **rDC20**. Pour les fils ACSR, **coreRadius** est facultatif

pour obtenir **gmr** par un calcul dépendant de la fréquence. **coreRadius** est de zéro pour les fils non-ACSR. **ratedCurrent** est le courant permanent admissible à 50 °C. **ratedCurrent** est nécessaire pour interpréter la sortie du calcul de répartition (power flaw), mais pas pour la solution du calcul de répartition elle-même. Les attributs **insulated**, **insulationMaterial** et **insulationThickness** s'appliquent principalement aux classes de câble dérivées, mais ils prennent également en charge les lignes secondaires triplex et le câble séparateur aérien. Les attributs **CableInfo** décrivent les propriétés supplémentaires des couches sur le noyau conducteur. Le **diameterOverCore** comprend un écran ou blindage conducteur et semi-conducteur; il convient qu'il soit du diamètre intérieur de l'isolant. Le **diameterOverInsulation** correspond au diamètre extérieur de la couche d'isolant ne comprenant pas les écrans extérieurs ou couches semi-conductrices. Le **diameterOverScreen** comprend l'isolant plus l'écran ou la couche semi-conductrice; il convient qu'il soit du diamètre intérieur du blindage ou de la gaine. Le **diameterOverJacket** correspond au diamètre extérieur du câble; il convient qu'il corresponde à la plus grande valeur de diamètre spécifiée, à l'exception des neutres concentriques.

Les attributs **tapeLap** et **tapeThickness** de **TapeShieldCableInfo** décrivent une bande mince recouvrant le câble, généralement en cuivre, présente dans les spires de chevauchement. Le **ConcentricNeutralCableInfo** a des attributs pour les conducteurs neutres extérieurs comportant généralement plusieurs fils de cuivre rigides. Le nombre de fils correspond à **neutralStrandCount**, le rayon est **neutralStrandRadius**, le rayon moyen géométrique est **neutralStrandGmr** et la résistance par fil est **neutralStrandRDC20**. Dans la mesure où ces fils sont trop petits, la résistance CA n'est généralement ni utilisée ni spécifiée. Le **diameterOverNeutral** correspond au diamètre sur les brins à neutre concentrique qui peut être identique à **diameterOverJacket**.

NOTE Le modèle illustré à la Figure 11 prend actuellement en charge les deux types de câble à conducteur simple les plus couramment utilisés sur les systèmes de distribution. De nombreux autres types de câbles utilisés dans les systèmes de transmission ou industriels tels que les câbles à 3 conducteurs, de canalisation, immergés et autres, ne sont pas pris en charge.

WireSpacingInfo identifie les données géométriques de ligne. Les attributs **phaseWireCount** et **phaseWireSpacing** se réfèrent à une mise en faisceau de sous-conducteurs, ce qui n'est pas courant pour les lignes de distribution, mais peut apparaître sur les lignes de transmission haute tension dans le modèle.

WirePosition définit les coordonnées horizontale (**xCoord**) et verticale (**yCoord**) de chaque fil sur la section de pôle ou de pylône, ou dans la tranchée/canalisation de câble, et **phase** identifie le fil de phase installé à cet endroit. Tous les trois attributs sont requis. La hauteur du fil par rapport au sol est **yCoord**, y compris tous les éventuels effets de flèche. Pour les câbles souterrains, **yCoord** est la profondeur d'enfouissement moyenne, saisie comme un nombre négatif. La position horizontale du fil, **xCoord**, est mesurée à partir d'une ligne de référence arbitraire mais constante. Les choix courants pour la référence horizontale sont l'axe des pôles et la position de fil la plus à gauche.

Tous les éventuels fils à une position **WirePosition** avec **phase=N** sont censés être mis en permanence à la terre. L'application peut éliminer ces conducteurs des matrices d'impédance et d'admittance par une réduction de Kron.

La valeur de l'attribut **WirePosition.phase** peut très souvent changer en fonction de l'installation sur le terrain. Par exemple, une ligne monophasée peut être utilisée sur la phase **A**, **B** ou **C**. Cela nécessite trois instances **WireSpacingInfo**, la seule différence résidant dans le fait qu'il existe trois valeurs différentes pour l'une des valeurs associées de **WirePosition.phase**, à savoir **A**, **B** ou **C**. La valeur de **WirePostion.phase** pour un fil de neutre ne changerait pas, il convient qu'elle soit toujours **N**. En résumé, un type de ligne monophasé nécessite 3 instances **WireSpacingInfo** pour couvrir toutes les possibilités de phasage. Un type de ligne diphasée nécessite 6 instances et un type de ligne triphasée nécessite également 6 instances.

4.4.3.3.1 Utilisation des impédances en séquence (cas équilibré)

Les impédances directes et homopolaires peuvent être transférées par les attributs **r**, **x**, **r0** et **x0** de **PerLengthSequenceImpedance** associé à une instance de **ACLineSegment**. Les attributs **bch**, **b0ch**, **gch** et **g0ch** ne sont généralement pas importants pour les lignes de distribution aériennes. Pour les trois phases, cela décrit une ligne triphasée équilibrée ou parfaitement transposée. Les attributs dans **PerLengthSequenceImpedance** sont exprimés en unités linéiques et il est donc nécessaire de multiplier leur valeur par l'attribut **length** du **Conductor**.

NOTE Cela équivaut aux attributs de **Wires::ACLineSegment**, qui sont pré-calculés pour la longueur totale du segment et sont définis sur chaque instance du segment. **PerLengthSequenceImpedance**, en revanche, est référençable et, de ce fait, peut être utilisé (à travers une association) par plusieurs instances de segment, diminuant ainsi la quantité de données transférées dans les échanges de données.

Si **ACLineSegment** a seulement une ou deux phases, un modèle équilibré peut encore être transféré à travers les attributs **r**, **x**, **r0** et **x0**. Cela représente une matrice d'impédances avec des éléments diagonaux complexes égaux, Z_s, et des éléments non diagonaux complexes égaux, Z_m. Pour une ligne monophasée, les attributs à transférer sont:

$$Z_1 = Z_0 = Z_s$$

Pour une ligne diphasée ou triphasée, les attributs à transférer sont:

$$Z_1 = Z_s - Z_m$$

$$Z_0 = Z_s + (n - 1)\, Z_m$$

où n est le nombre de phases. À la réception de **r**, **x**, **r0** et **x0**, la matrice des impédances en diphasé ou triphasé est construite à partir de:

$$Z_s = (Z_0 + (n - 1)\, Z_1) / n$$

$$Z_m = (Z_0 - Z_1) / n$$

Il convient que le **ACLineSegment** de référencement ait des instances associées de **ACLineSegmentPhase**, et affecte les valeurs appropriées de **phase** pour indiquer les phases réellement présentes, telles que **A**, **B**, **C**, **s1** ou **s2**. Il convient que le neutre, **N**, n'apparaisse pas, car tout conducteur neutre doit avoir été incorporé dans le retour par la terre lorsque les impédances en séquence sont utilisées.

Pour les câbles de distribution souterrains, les impédances en séquence sont également appropriées, y compris les attributs **bch** et **b0ch**.

4.4.3.3.2 Utilisation des impédances de phase (cas déséquilibré)

Les paramètres calculés de la matrice peuvent être transférés par référencement d'une instance **PerLengthPhaseImpedance**, en lieu et place du modèle physique décrit en 4.4.3.3.3. Un modèle de matrice est utile lorsque:

- l'application cible n'a aucun moyen de calculer les paramètres à partir des données physiques;
- les données physiques sous-jacentes ne sont pas disponibles directement;
- il est nécessaire de réaliser une adaptation la plus étroite possible des paramètres de ligne déséquilibrée.

Pour une ligne diphasée, avec le neutre déjà réduit, les matrices Z et Y seront carrées 2x2, mais, en raison de la symétrie, il n'y aura que 3 éléments de matrice uniques. Cela conduit à trois instances associées de **PhaseImpedanceData** avec stockage en colonne:

- **sequenceNumber** = 1 pour la ligne 1, colonne 1 de la matrice;
- **sequenceNumber** = 2 pour la ligne 2, colonne 1 de la matrice;
- **sequenceNumber** = 3 pour la ligne 2, colonne 2 de la matrice.

Cette instance de **PerLengthPhaseImpedance** pourrait être référencée à partir d'un **ACLineSegment** ayant les phases **AB**, **AC** ou **BC**. La ligne 1 correspond toujours à la première phase présente et la ligne 2 correspond toujours à la deuxième phase présente, suivant l'ordre **A**, **B**, **C, s1**, **s2**, **N**. Si les fils de phase sont installés dans différentes positions, de sorte qu'il convient de placer **C** (par exemple) dans la ligne 1, une nouvelle instance de **PerLengthPhaseImpedance** est requise.

4.4.3.3.3 Utilisation des paramètres physiques

Pour les lignes aériennes, un modèle physique peut être transféré par référence à une instance de **WireSpacingInfo**, qui est associée à des classes supplémentaires montrées à la Figure 11. Cela viendra à l'appui du calcul d'une matrice des impédances de phase non équilibrées par l'utilisation des équations de Carson ou d'une méthode équivalente de prise en charge du retour par la terre. Par exemple, supposons qu'il y ait trois fils de phase, plus un fil de neutre de taille différente, sur un poteau à console horizontale. Cela exige une instance de **WireSpacingInfo**, quatre instances de **WirePosition** qui décrivent les quatre positions des conducteurs, et deux instances de **WireInfo** décrivant les types des fils de phase et du neutre. L'attribut **length** de **Conductor** doit être utilisé, et plusieurs **ACLineSegment** se référeront typiquement à la même instance de **WireSpacingInfo**.

Les instances de **WirePosition** ont un attribut **phase** qui correspond à **ACLineSegmentPhase.phase**. Pour les calculs, il convient de fournir l'attribut résistance de **WireInfo** pour la fréquence fondamentale du réseau et à la température de fonctionnement souhaité du fil.

Un câble à neutre concentrique monophasé requiert une instance de **ConcentricNeutralCableInfo**, un **WireSpacingInfo** avec l'attribut **isCable**=true, et une instance de **WirePosition** pour spécifier la profondeur d'enfouissement. Un câble blindé triphasé, avec un conducteur neutre nu, exigerait une instance de **TapeShieldCableInfo,** un **WireSpacingInfo** avec l'attribut **isCable**=true et quatre **WirePosition** pour les trois phases et le neutre. Il y aurait également un **WireInfo** pour le neutre.

4.4.3.4 Transformateurs de distribution

4.4.3.4.1 Modèle électrique

La Figure 12 montre les classes qui modélisent les instances du transformateur de puissance. Elles peuvent utiliser exclusivement le paquetage **Wires** pour définir les paramètres d'impédance ou utiliser les classes **AssetInfo**, détaillées à la Figure 13, pour définir une bibliothèque de types de transformateur.

NOTE La présente partie de la CEI 61968 (documentant le modèle DCIM11) et l'édition correspondante de la CEI 61970-301 (documentant le modèle CIM15 de base) reflètent des modèles consolidés de transformateur pour la transmission et la distribution (T&D). Par conséquent, les classes issues de la CEI 61968-11:2010 (1ère édition) ont été légèrement modifiées et déplacées du paquetage de modèle **IEC61968::WiresExt** au paquetage de modèle **IEC61970::Wires** du CIM de base, CEI 61970-301 (5ème édition).

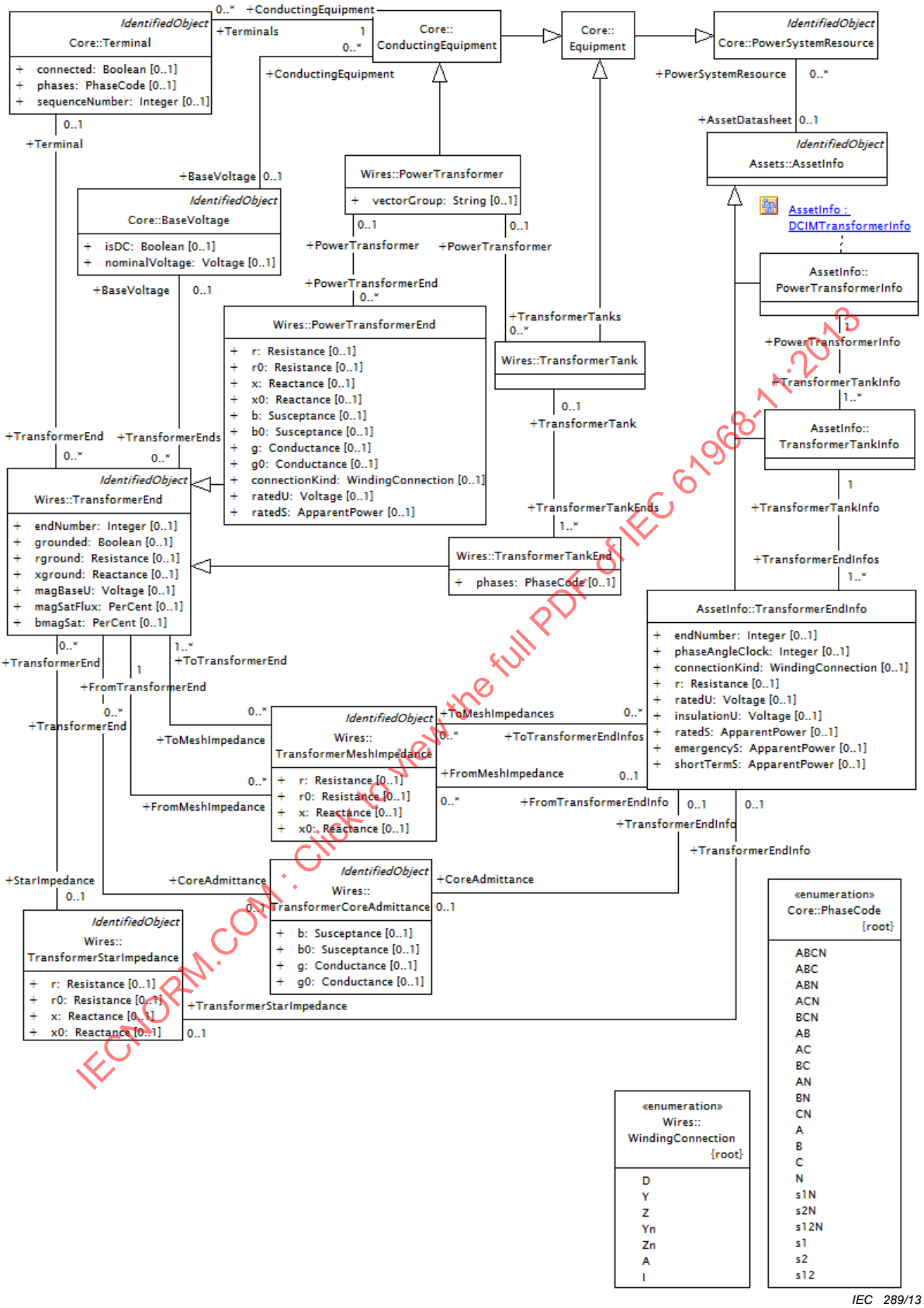


Figure 12 – Modèle de connectivité des transformateurs DCIM

PowerTransformer est l'instance de niveau le plus élevé pour un transformateur de puissance, comportant une cuve (tank) triphasée ou éventuellement différentes cuves monophasées. Il est issu de **ConductingEquipment**, et dispose par conséquent de bornes **Terminal** associées

avec les valeurs de l'attribut **phases** à chaque point de connexion d'enroulement. Dans le CIM, un enroulement de transformateur est désigné comme une "extrémité".

Lorsqu'il est constitué de différentes cuves, un **PowerTransformer** a souvent été appelé "batterie de transformateurs", et le CIM prend en charge la modélisation avec ou sans cuves. Dans les systèmes de distribution, les régulateurs indépendants de tension de phase et les transformateurs à connexion étoile/delta ouverte constituent deux exemples qui nécessitent une modélisation au niveau cuve. Au niveau de la transmission, les batteries de transformateurs THT peuvent aussi contenir des transformateurs monophasés, qui peuvent ne pas être identiques, notamment lorsqu'un élément de rechange est en service. L'attribut **vectorGroup** pour le relayage de protection est dérivé des connexions d'enroulements internes et des angles de phase. Il utilise la nomenclature de la CEI 60076-1 pour décrire le nombre des enroulements qui peuvent être compris dans la cuve.

Lorsqu'il est utilisé, le **TransformerTank** doit être associé à un **PowerTransformer**. Il hérite de **Equipment**, et non de **ConductingEquipment**. La cuve peut avoir un **TransformerTankInfo** associé issu du paquetage **AssetInfo**, pour la modélisation de feuille de données de biens, décrite de manière plus détaillée ultérieurement à la Figure 13 et en 4.4.3.4.2 Etant donné que les essais de transformateur sont réalisés sur les cuves, les feuilles de données sont fondamentalement associées aux cuves. Lorsque le modèle n'utilise pas les cuves, le **PowerTransformer** peut encore avoir une association à **PowerTransformerInfo**, pour la modélisation de feuille de données de biens. Dans les deux cas, les données résident effectivement dans les instances de **TransformerEndInfo**, associées à un **TransformerTankInfo**.

Deux méthodes permettent de référencer les feuilles de données de biens de transformateur et un profil peut adopter l'une ou l'autre:

- (**PowerSystemResource**–**AssetInfo**) Utiliser l'extrémité d'association **AssetDatasheet** de **PowerTransfomer** ou **TransformerTank** pour référencer respectivement **PowerTransformerInfo** ou **TransformerTankInfo**, ou
- (**PowerSystemResource**–**Asset**–**AssetInfo**) Instancier la classe **Asset**, dans laquelle l'extrémité d'association **AssetInfo** référence le **PowerTransformerInfo** ou le **TransfomerTankInfo**, et plusieurs **PowerSystemResource** référencent chaque **PowerTransformer** ou **TransformerTank** en utilisant la feuille de données considérée.

TransformerEnd, appelé TransformerWinding dans les précédentes versions du CIM et qui était un **ConductingEquipment**, n'est plus un **ConductingEquipment**, mais il dispose d'une borne **Terminal** associée avec des informations relatives au phasage. Les attributs **magBaseU**, **magSatFlux** et **bmagSat** représentent la saturation du noyau, généralement modélisée à pas plus d'une des extrémités. Les autres attributs d'instance définissent les options de mise à la terre:

- solidement mis à la terre: **grounded** = true, **rground** = 0, **xground** = 0;
- mis à la terre par une impédance: **grounded** = true, **rground** ≥ 0, **xground** ≥ 0;
- non mis à la terre: **grounded** = false.

Il convient de ne pas instancier directement **TransformerEnd**; il convient d'instancier l'un de ses deux descendants:

- **PowerTransformerEnd**, si la modélisation au niveau de la cuve n'est pas utilisée. Spécifier les valeurs des attributs **ratedU** et **ratedS** pour les données des caractéristiques nominales d'enroulement, et **connectionKind** pour les connexions étoile, delta ou autre type de connexion.
- **TransformerTankEnd**, si la modélisation au niveau de la cuve est utilisée. Les valeurs relatives à la connexion de l'enroulement et aux caractéristiques nominales proviennent d'une instance de **TransformerTankEndInfo**, ce qui signifie que le paquetage **AssetInfo** est

requis pour la modélisation au niveau de la cuve. L'attribut **phases** prend en charge la modélisation de cuve par phase.

En utilisant **PowerTransformer** et **PowerTransformerEnd**, trois méthodes permettent de spécifier les paramètres d'impédance en utilisant uniquement le paquetage de modèle **IEC61970::Wires** de la CEI 61970-301:

- Utiliser les attributs **r**, **x**, **r0**, **x0**, **b**, **b0**, **g** et **g0** de **PowerTransformerEnd** pour spécifier les paramètres d'impédance pi. Il s'agit de l'option la plus compatible avec les précédentes versions de la CEI 61970-301, dont cette norme décrit certaines conventions importantes..
- Utiliser une **TransformerStarImpedance** associée pour chaque **PowerTransformerEnd**, comportant un équivalent en étoile (également parfois appelé équivalent en T ou en étoile). Cela peut se révéler exact du point de vue mathématique jusqu'à trois extrémités (enroulements). Cependant, des valeurs d'attribut négatives peuvent se produire en présence de trois extrémités. En variante, utiliser une **TransformerCoreAdmittance** associée à une des **PowerTransformerEnd**, représentant le courant d'excitation et les pertes dans le fer. Il peut s'agir de l'enroulement de plus basse tension (c'est-à-dire le plus proche du noyau) ou de l'enroulement ayant été réellement soumis à un essai à vide, s'il est connu. La tension de référence applicable à toutes les valeurs d'attribut, exprimées en unités de Ohms ou de Siemens, doit être **ratedU** pour l'extrémité à laquelle est connectée l'impédance ou l'admittance.
- Utiliser une **TransformerMeshImpedance** associée à chaque combinaison de paires **PowerTransformerEnd**. Il doit y en avoir (numberEnds-1) × numberEnds / 2. Par exemple, une **TransformerMeshImpedance** entre deux extrémités, trois d'entre elles entre trois extrémités, six d'entre elles entre quatre extrémités, etc. Le modèle de maille présente les avantages suivants: (a) il est exact d'un point de vue mathématique pour plus de trois extrémités, (b) il n'a pas de valeurs d'attribut négatives, et (c) il correspond plus directement aux données d'essai de court-circuit de transformateur. La tension de référence doit être **ratedU** de **FromTransformerEnd** (à noter que l'autre extrémité plus proche a toujours une tension nominale différente). En variante, utiliser une **TransformerCoreAdmittance** telle que décrite pour l'équivalent en étoile.

4.4.3.4.2 Modèle de feuilles de données

La Figure 13 montre les classes qui permettent l'échange de modèles de feuilles de données (datasheet) des transformateurs, parfois appelées «codes de transformateur» dans les applications.

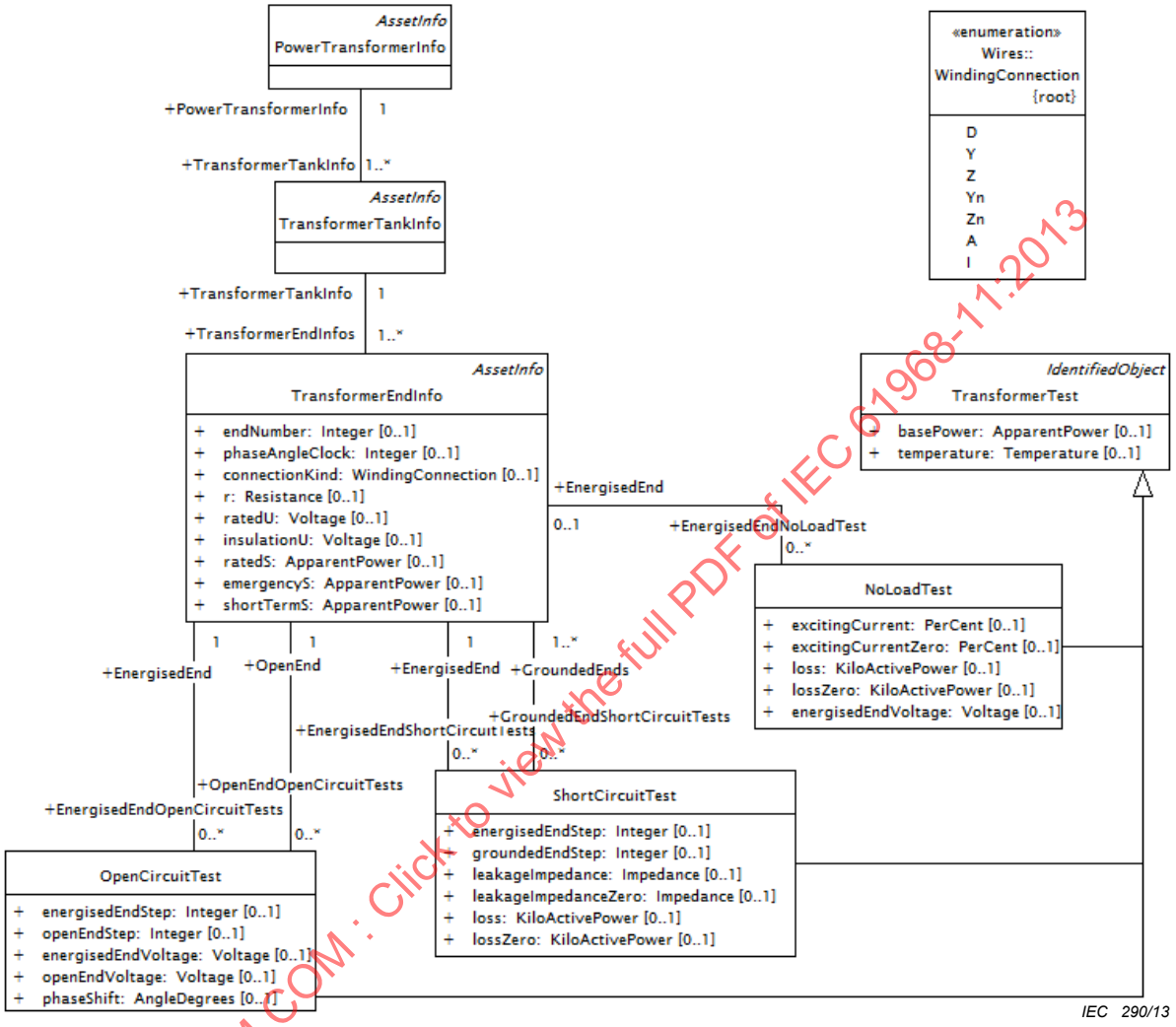


Figure 13 – Modèle de feuilles de données des transformateurs DCIM

La classe principale est **TransformerEndInfo**, qui contient les données relatives aux caractéristiques nominales et aux connexions de l'enroulement de transformateur correspondant. Les données relatives aux caractéristiques nominales comprennent **ratedU**, **ratedS**, **shortTermS**, **emergencyS** et **insulationU**. L'attribut **r** correspond à la résistance CC de l'enroulement.

Les données relatives aux connexions sont contenues dans les attributs **connectionKind**, **phaseAngleClock** et **endNumber**. L'attribut **endNumber** est l'ordre de l'enroulement dans le **vectorGroup** du **PowerTransformer** de référencement et il est utilisé pour faire correspondre les données à **TransformerEnd.endNumber**. Le **Terminal.sequenceNumber** associé peut également utiliser le même **TransformerEnd.endNumber**, mais cela n'est pas exigé. Par convention, le **endNumber** commence généralement à 1 pour l'enroulement de tension la plus élevée, et tous les autres enroulements sont numérotés dans l'ordre descendant des caractéristiques nominales de tension. Certains transformateurs ont plusieurs enroulements ayant la même valeur **ratedU**, des transformateurs diviseurs de secondaire (split-secondary transformer) fournissant juste un exemple et ils doivent avoir des **endNumber** différents.

L'énumération **WindingConnection,** utilisée comme type pour l'attribut **connectionKind**, comprend la nomenclature normalisée **D**, **Y**, **Z**, **Yn** et **Zn** pour décrire les connexions delta, étoile (en Y), zigzag et à neutre dans les groupes vectoriels des transformateurs triphasés. **A** est utilisé pour l'enroulement d'autotransformateur commun et **I** pour un enroulement de transformateur monophasé.

L'attribut **endNumber** permet d'utiliser la bibliothèque de feuilles de données et de l'actualiser avec une seule association à **PowerTransformerInfo** ou **TransformerTankInfo**. Le **TransformerTankInfo** et le **PowerTransformerInfo** fournissent deux chemins de navigation vers **TransformerEndInfo**. Avec **PowerTransformerInfo**, il peut être nécessaire de créer un **TransformerTankInfo** artificiel pour le transfert de modèle car les feuilles de données se situent au niveau de la cuve. Une instance de **PowerTransformerInfo** fait réellement référence à un ensemble de une ou de plusieurs cuves.

TransformerTankInfo est référencé par **TransformerTank** (**PowerSystemResource–AssetInfo** hérité) et **PowerTransformerInfo** (directement). Il sert principalement à organiser les données en une bibliothèque avec un point d'entrée. **TransformerTankInfo** rassemble les associations à **TransformerEndInfo**, ce qui se généralise à n'importe quel nombre d'enroulements.

Deux méthodes permettent de spécifier les paramètres d'impédance dans un modèle de feuille de données:

- Associer **TransformerEndInfo** aux classes **TransformerMeshImpedance**, **TransformerStarImpedance** et **TransformerCoreAdmittance** du paquetage de modèle **IEC61970::Wires** de la CEI 61970-301. Ces associations ont été illustrées à la Figure 12 et décrites en 4.4.3.4.1.
- Utiliser **TransformerTest** et ses trois classes descendantes, illustré à la Figure 13. Chaque application est chargée de convertir les données d'essai en équivalent en maille, équivalent en étoile ou autre modèle d'impédance électrique.

TransformerTest est la classe parente pour toutes les classes d'essai de transformateur, les attributs **basePower** et **temperature** étant applicables à tous les essais. **basePower** est essentiel pour convertir les valeurs des feuilles de données en impédance ou admittance à la bonne tension de référence.

Dans le cas de **NoLoadTest,** ses attributs **excitingCurrent**, **excitingCurrentZero**, **loss** et **lossZero** sont tous mesurés sur l'enroulement **EnergisedEnd** et fournissent les données de base pour une branche d'admittance centrale. Les données d'essai directes et homopolaires peuvent être rapportées dans la même instance de **NoLoadTest**. Cet essai est généralement réalisé en appliquant la tension nominale à l'enroulement sous tension, mais différentes valeurs peuvent être spécifiées dans **energisedEndVoltage**, et plusieurs essais peuvent être prévus pour définir les paramètres de saturation du noyau.

ShortCircuitTest est réalisé en faisant circuler le courant nominal à travers l'enroulement **EnergisedEnd**, un ou plusieurs enroulements **GroundedEnds** étant mis en court-circuit. Il fournit les données de base pour le circuit équivalent en maille ou en étoile. Si les essais ont été réalisés pour différents réglages de prise, les valeurs de prise sont spécifiées dans **energisedEndStep** et **groundedEndStep**. Les données d'essai directes et homopolaires peuvent être rapportées dans la même instance de **ShortCircuitTest.** Les résistances CA dérivées de **loss** et de **lossZero** vont vraisemblablement différer des résistances CC des enroulements, **TransformerEndInfo.r**, qui sont obtenues à partir d'un essai distinct.

Les attributs **openEndVoltage** et **phaseShift** de **OpenCircuitTest** sont mesurés sur un enroulement ouvert simple lorsque la **energisedEndVoltage** est appliquée à **EnergisedEnd**. Les réglages de prise pour les deux enroulements sont spécifiés dans **energisedEndStep** et **openEndStep**. Les essais sont réalisés pour vérifier le rapport d'enroulements, les connexions d'enroulement et la polarité d'enroulement du transformateur. Ils ne sont généralement pas nécessaires pour déterminer les paramètres d'impédance électrique.

4.4.3.4.3 Modèle de changeur de prise

La Figure 14 montre les classes utilisées pour modéliser un régulateur de distribution de tension déséquilibrée. Un **RatioTapChanger** est associé à un **TransformerEnd**. Chaque régulateur utilise une commande locale autonome et, de ce fait, **RegulationSchedule** et **TapSchedule** (présents dans la CEI 61970-301 et généralement utilisés pour le transport) ne sont pas utilisés dans le cas présent. Certains attributs importants de ces classes ont été copiés dans **DistributionTapChanger**. Les régulateurs d'angle de phase et les courbes de variations ne sont également pas utilisés d'une manière générale dans les systèmes de distribution.

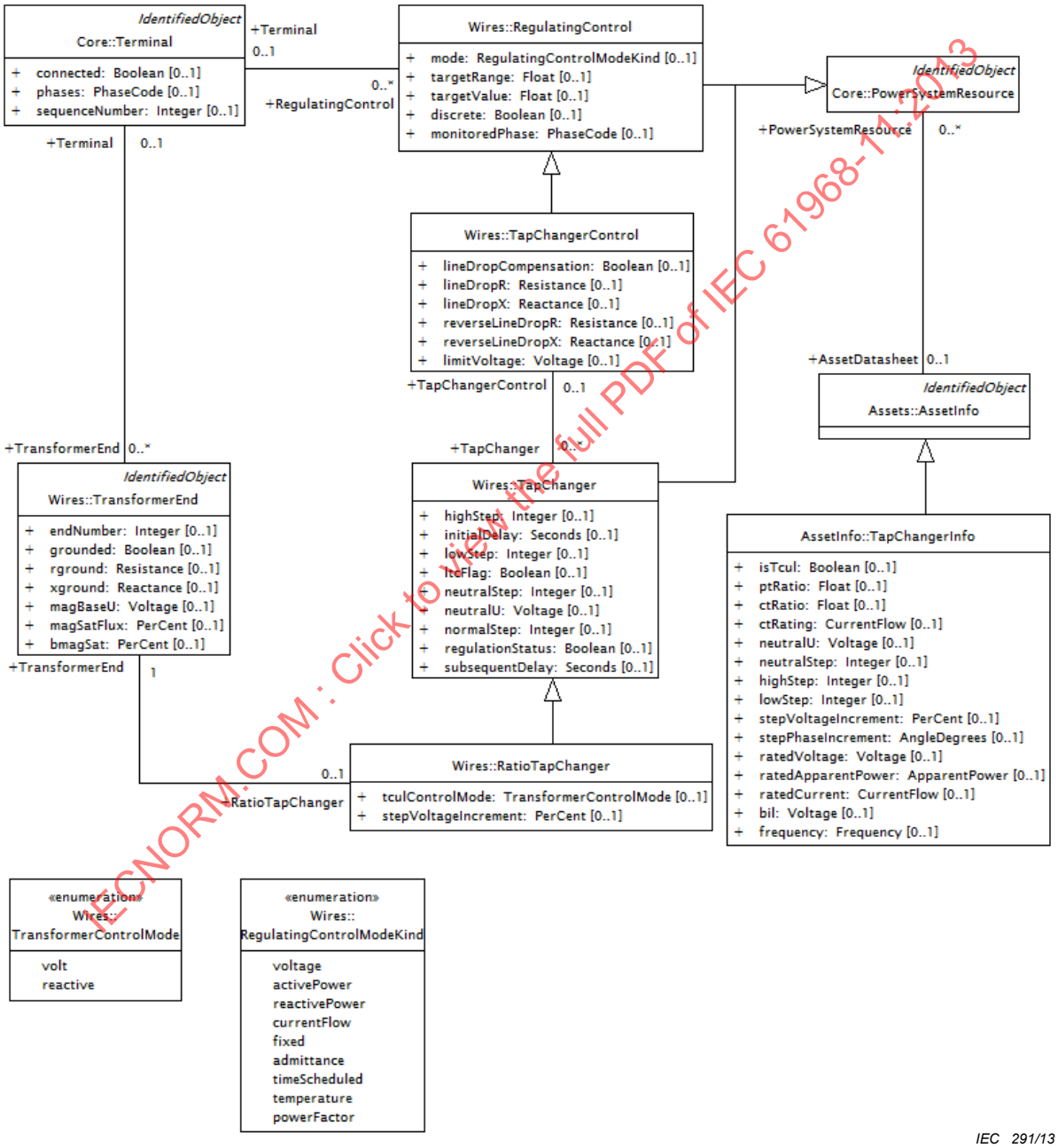


Figure 14 – Modèle de changeur de prise DCIM

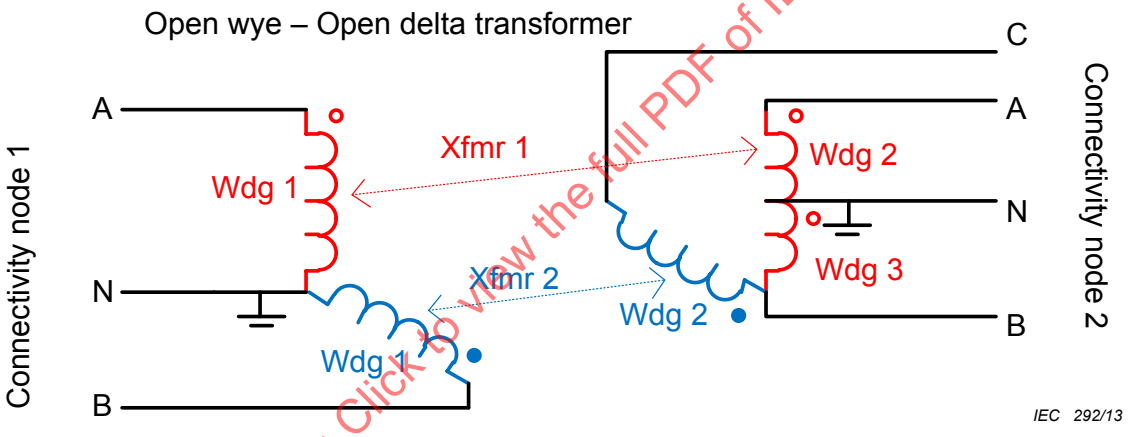
Un régulateur de tension de ligne triphasée comporte habituellement trois régulateurs indépendants pour contribuer à corriger le déséquilibre de la tension électrique. Les régulateurs sont habituellement connectés en étoile. Le modèle démarre avec un **PowerTransformer** contenant trois **TransformerTank** et un total de six **TransformerTankEnd**. Il y

aura trois instances de **RatioTapChanger**, associées chacune à un **TransformerTankEnd** différent. Il convient que l'attribut **RegulatingControl.monitoredPhase** soit compris dans les **Terminal.phases** associées au **PowerTransformer.** Les positions des prises, et parfois les autres attributs, ne seront pas les mêmes dans chaque phase du régulateur. Un régulateur en triangle ouvert est également assez courant. Il consiste en deux régulateurs monophasés reliés phase à phase dans une cuve, avec une capacité partielle de corriger le déséquilibre de tension électrique.

Un régulateur de tension de poste triphasée change habituellement toutes les trois prises ensemble, sans capacité de corriger le déséquilibre de tension électrique. Dans ce cas, la modélisation au niveau de la cuve n'est pas nécessaire et le modèle peut comprendre un **PowerTransformer** avec deux **PowerTransformerEnd**s Il y aurait juste un seul **RatioTapChanger** associé à un **PowerTransformerEnd**. L'attribut **RegulatingControl.monitoredPhase** peut être **A**, **B** ou **C** si le transformateur de potentiel est raccordé phase-terre. Il peut aussi être **AB**, **AC** ou **BC** pour les transformateurs de potentiel phase-phase. Typiquement, un seul transformateur de potentiel commande ce type de régulateur.

4.4.3.4.4 Exemple de transformateur de distribution

Le transformateur dans la Figure 15 est une batterie étoile ouverte/triangle ouvert, qui est utilisée pour alimenter un service triphasé peu coûteux à des clients plus petits.



Légende

Anglais	Français
Open wye – Open delta transformer	Triangle ouvert – Transformateur en étoile ouvert
Connectivity node 1	Nœud de connectivité 1
Connectivity node 2	Nœud de connectivité 2
Wdg 1	Wdg 1 (c'est-à-dire, Enroulement 1)
Xfmr 1	Xfmr 1 (c'est-à-dire, Transformateur 1)

Figure 15 – Exemple de transformateur de distribution pouvant être modélisé par le DCIM

Le Tableau 1 montre certaines des valeurs d'attribut importantes pour cet exemple. Il nécessite une modélisation au niveau de la cuve.

Tableau 1 – Connexions de batterie de transformateur en étoile ouverte / triangle ouvert

Cuve de transformateur (Transformer Tank)	Enroulement de cuve (Transformer TankEnd)	ratedU	ratedS	Type de connexion	déphasage	Phases d'enroulement de cuve (Transformer TankEnd)
Xfmr 1	Wdg 1	7 200	100e3	I	0	AN
	Wdg 2	120	50e3	I	0	AN
	Wdg 3	120	50e3	I	6	BN
Xfmr 2	Wdg 1	7 200	50e3	I	0	BN
	Wdg 2	240	50e3	I	0	BC

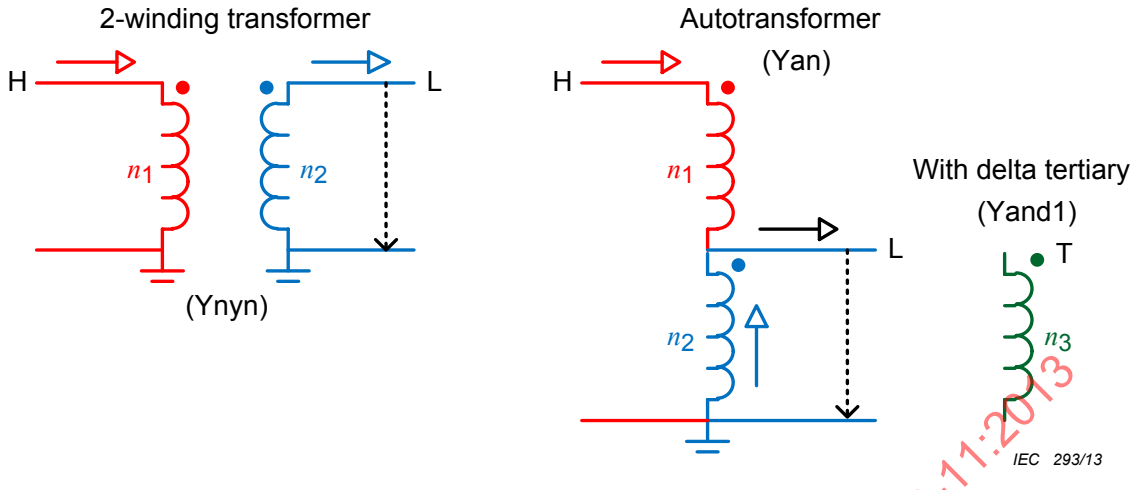
Un angle de phase selon la notation horaire "6" indique que l'enroulement Wdg 3 est en réalité de N vers B, plutôt que de B vers N. À travers les bornes **Terminal**, le nœud de connectivité **ConnectivityNode** 1 aura la présence des phases **ABN**. Un autre équipement raccordé, tel qu'un segment de ligne, pourrait ajouter une phase **C**. **ConnectivityNode** 2 aura la présence des phases **ABCN**. La "patte d'éclairage" (Xfmr 1) a habituellement une caractéristique nominale différente de celle de la "patte alimentation" (Xfmr 2). Cela signifie que les affectations de phases et de caractéristiques nominales à la cuve pourraient être ambiguës et il serait donc nécessaire de les spécifier sur **TransformerTankEnd**.

4.4.3.4.5 Exemple d'autotransformateur

Un autotransformateur est réalisé en connectant deux enroulements de transformateur en série afin d'établir une connexion métallique entre les deux niveaux de tension. Cette configuration présente l'avantage de réduire les coûts car la caractéristique nominale MVA est supérieure à celle applicable aux deux mêmes enroulements connectés comme un transformateur conventionnel. Les autotransformateurs sont également plus efficaces et présentent une chute de tension moindre. Le principal inconvénient réside probablement dans le courant de court-circuit plus élevé qui circule dans les systèmes totalement développés dans la mesure où les autotransformateurs présentent une impédance plus faible. Les deux applications communes sont:

- transformations entre deux niveaux de très haute tension (THT) dans un poste dont les réductions de coût représentent un facteur important pour les rapports d'enroulements jusqu'à environ 2:1.
- régulateurs de tension de ligne sur les départs, où l'enroulement de régulation (abaisseur de tension/élévateur de tension) est connecté en auto.

La Figure 16 montre un transformateur à deux enroulements vers la gauche, avec une connexion d'essai de court-circuit sur la borne d'enroulement L. La totalité du courant est transformé magnétiquement et le rapport d'enroulements est n_1:n_2. Vers la droite, les deux enroulements sont connectés selon la configuration d'un autotransformateur. Le courant de couleur rouge circule directement vers la connexion d'essai de court-circuit sur la borne d'enroulement L. De plus, le courant de couleur bleue transformé magnétiquement contribue également au courant de court-circuit qui est supérieur à celui du transformateur à deux enroulements. Le rapport d'enroulements de l'autotransformateur est (n_1+n_2):n_2. L'enroulement H est parfois appelé enroulement série (S) dans un autotransformateur, et l'enroulement L est parfois appelé enroulement commun (C) dans un autotransformateur.



Légende

Anglais	Français
2-winding transformer	Transformateur à deux enroulements
Autotransformer	Autotransformateur
With delta tertiary	Avec tertiaire en triangle

Figure 16 – Exemple de transformateur à deux enroulements connecté comme un autotransformateur

Les groupes vectoriels de la CEI 60076-1 définissent les caractéristiques de mise à la terre et de déphasage de chaque enroulement qui sont importantes pour le relayage de protection, la mise en parallèle et d'autres applications – la Figure 16 illustre cela entre parenthèses. "Ynyn" désigne un transformateur à deux enroulements tous deux mis à la terre en étoile. (L'impédance neutre extérieure peut encore être ajoutée mais elle n'est pas répertoriée dans le groupe vectoriel). L'autotransformateur peut également être désigné "Ynyn", dans la mesure où il a les mêmes caractéristiques de connexion neutre et de déphasage. Cependant, certains fournisseurs de transformateur utilisent "A" ou "a" pour indiquer un enroulement d'autotransformateur. La Figure 16 montre le groupe vectoriel "Yan" pour un autotransformateur. Même si la borne H a un chemin conducteur vers le neutre passant par l'enroulement L, l'enroulement H proprement dit n'est pas connecté au neutre. Selon la CEI 60076-1, l'enroulement de la plus haute tension est indiqué en majuscules dans le groupe vectoriel tandis que tous les autres enroulements sont en minuscules. Cela signifie dans la pratique que la lettre "a" pour un autotransformateur serait toujours en minuscule.

De nombreux autotransformateurs ont un tertiaire en triangle, tel qu'indiqué à droite sur la Figure 16. Le groupe vectoriel serait "Yand1", où 1 indique un décalage de 30° (1 heure) par rapport à l'enroulement H. La valeur d'impédance de maille Z_{HT} est affectée par cette connexion automatique mais pas la valeur d'impédance de maille Z_{LT}. L'effet sur Z_{HT} n'est pas présenté dans le cas présent mais il peut être trouvé dans plusieurs références techniques.

Les conversions des données relatives aux deux enroulements (à gauche sur la Figure 16) en rapport d'enroulements d'autotransformateur (N), caractéristique nominale voltampère (S_{auto}) et impédance de maille (Z_{auto}) (à droite sur la Figure 16) sont:

$$N = 1 + n_1 / n_2$$

$$S_{auto} = S_{2\text{-wdg}} \, [N / (N - 1)]$$

$$Z_{auto} = Z_{2\text{-wdg}} \, [(N - 1) / N]^2$$

Exprimée par unité, l'impédance convertie de l'autotransformateur ($Z_{\text{pu-auto}}$) est:

$$Z_{\text{pu-auto}} = Z_{\text{pu-2-wdg}} / N$$

Par exemple, soit un transformateur à deux enroulements de 115/115 kV, caractéristique nominale de 100 MVA, avec une impédance de 10 % sur 100 MVA. Le rapport d'enroulements est 1:1. Quel que soit l'enroulement considéré, l'impédance de court-circuit est de 13,225 Ω. Connecté comme un autotransformateur, 230/115 kV, le rapport d'enroulements est 2:1 (N=2) et la caractéristique nominale est 200 MVA. Considéré du point de vue de la borne 115 kV, l'impédance de court-circuit est de 3,306 25 Ω, ce qui représente 5 % pour la nouvelle caractéristique nominale de 200 MVA. Les coûts relatifs au noyau de fer et à l'enroulement de cuivre représentent la moitié de ce qu'ils seraient pour un transformateur à deux enroulements conventionnels ayant la même caractéristique nominale. Le coût total est quelque peu supérieur à 50 %, car les conducteurs de l'enroulement L doivent transporter plus de courant et l'enroulement H doit être isolé pour une tension plus élevée. La feuille d'essai de cet autotransformateur montrerait un rapport de transformation de 230 / 115 kV, une caractéristique nominale de 200 MVA et une impédance de court-circuit de 5 %.

Il est courant de modéliser un autotransformateur comme un transformateur à deux enroulements conventionnel, en utilisant les données de la feuille d'essai sans tenir compte du fait que les enroulements sont réellement connectés en série. Cependant, dans certains cas la différence revêt une importance particulière, comme par exemple dans le cadre d'une modélisation de noyau plus précise ou une modélisation plus précise de l'impédance par rapport à la caractéristique de la prise. Il est possible de dériver le modèle d'autotransformateur physique de la Figure 16, si les enroulements série et commun ont été identifiés.

Dans le CIM, il convient de modéliser un autotransformateur avec les données de configuration à deux enroulements conventionnelle relatives aux impédances, admittances et caractéristiques nominales, comme cela est généralement indiqué sur les rapports d'essai d'autotransformateur. Chaque enroulement physique a un **PowerTransformerEnd** ou **TransformerTankEnd** correspondant dans le CIM. Il est facultatif de spécifier la connexion de l'autotransformateur avec une valeur d'attribut **A** pour **connectionKind** sur l'extrémité commune, et avec "an" figurant dans le cadre du **PowerTransformer.vectorGroup** pour l'extrémité considérée. L'extrémité série doit alors avoir la valeur d'attribut **Y** pour **connectionKind**. Il convient que les **endNumber** série et commun soit 1 et 2, respectivement. L'application de réception peut alors dériver le modèle d'autotransformateur physique si cela se révèle nécessaire. Afin d'ignorer les connexions de l'autotransformateur dans le modèle, spécifier **Yn** pour **connectionKind** sur les extrémités série et commune; aucune restriction ne s'applique aux **endNumber**s. L'attribut **connectionKind** apparaît sur **TransformerEndInfo** si l'on utilise la modélisation au niveau de la cuve et sur **PowerTransformerEnd** si l'on n'utilise pas la modélisation au niveau de la cuve A noter que **A**, **Y**, **D** et **Z** sont toujours en majuscules dans **connectionKind**, mais si le **endNumber** est supérieur à 1, il convient qu'ils soient en minuscules dans le **vectorGroup**.

4.4.3.5 Équipement auxiliaire

La présente édition de la CEI 61968-11 comprend le moyen de modéliser l'équipement auxiliaire par le paquetage de modèle **IEC61970::Wires** de la CEI 61970-301, illustré à la Figure 17.

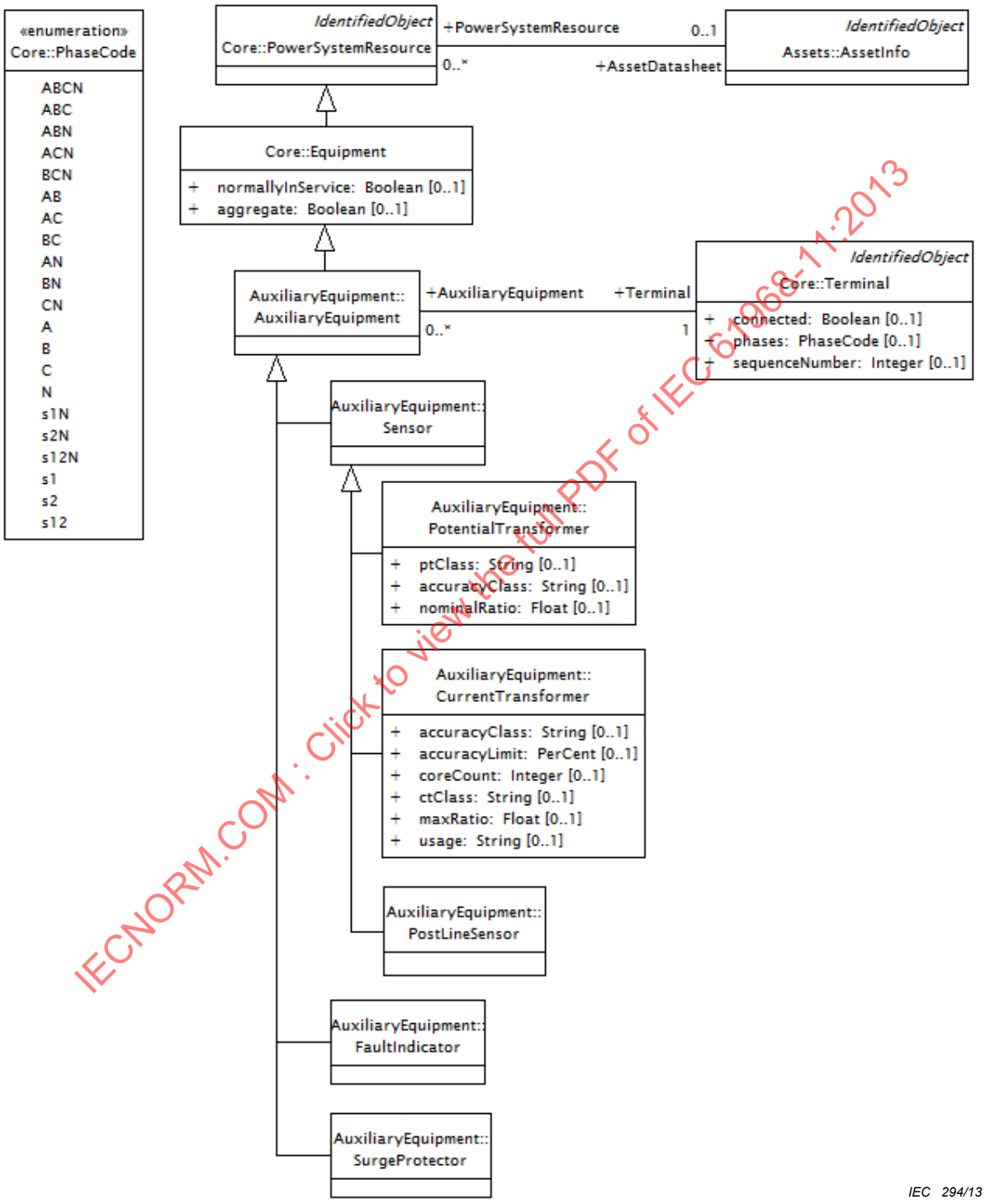


Figure 17 – Equipement auxiliaire DCIM

L'association de tout **AuxiliaryEquipment** à un **Terminal** représente un ajout important qui permet de positionner les dispositifs auxiliaires dans le modèle de connectivité du réseau électrique. Par exemple, il est désormais possible de modéliser de manière explicite les

transformateurs de mesure, de les positionner sur les schémas unifilaires avec le **ConductingEquipment** habituel et d'en déduire les **Measurement** qu'ils produisent réellement. Cette sorte d'échange de données concerne de manière générale l'importation de modèles de poste définis avec des modèles de données autres que le CIM.

De même, il est désormais possible d'échanger le "placement" dans le réseau des instances de **FaultIndicator**, comme cela s'applique pour les opérations de localisation des pannes, d'isolation et de restauration del'alimentation qui facilitent l'acheminement des équipages vers la partie du réseau (**ACLineSegment**) où le risque de panne est le plus susceptible de se produire.

L'ajout de la classe **Sensor** fournit le point d'extensibilité à des sortes d'équipement de détection autres que les transformateurs de mesure et **PostLineSensor**.

NOTE La présente partie de la CEI 61968 (documentant DCIM11) ne prend pas en compte la modélisation des feuilles de données de tous ces dispositifs auxiliaires, c'est-à-dire que la Figure 18 montre la classe **AssetInfo** sans spécialisations. Il est prévu d'ajouter les classes correspondantes de modèle de biens dans la prochaine édition afin de permettre une description plus détaillée de ces biens venant à l'appui des échanges de données exigés par la CEI 61968-3 et la CEI 61968-4.

4.4.4 Modèle Customers (clients)

Le paquetage **Customers** présente les classes, illustrées à la Figure 18, nécessaires à l'intégration par les entreprises des systèmes d'informations relatives aux clients et de facturation ainsi que les informations qu'ils échangent avec d'autres systèmes d'entreprise.

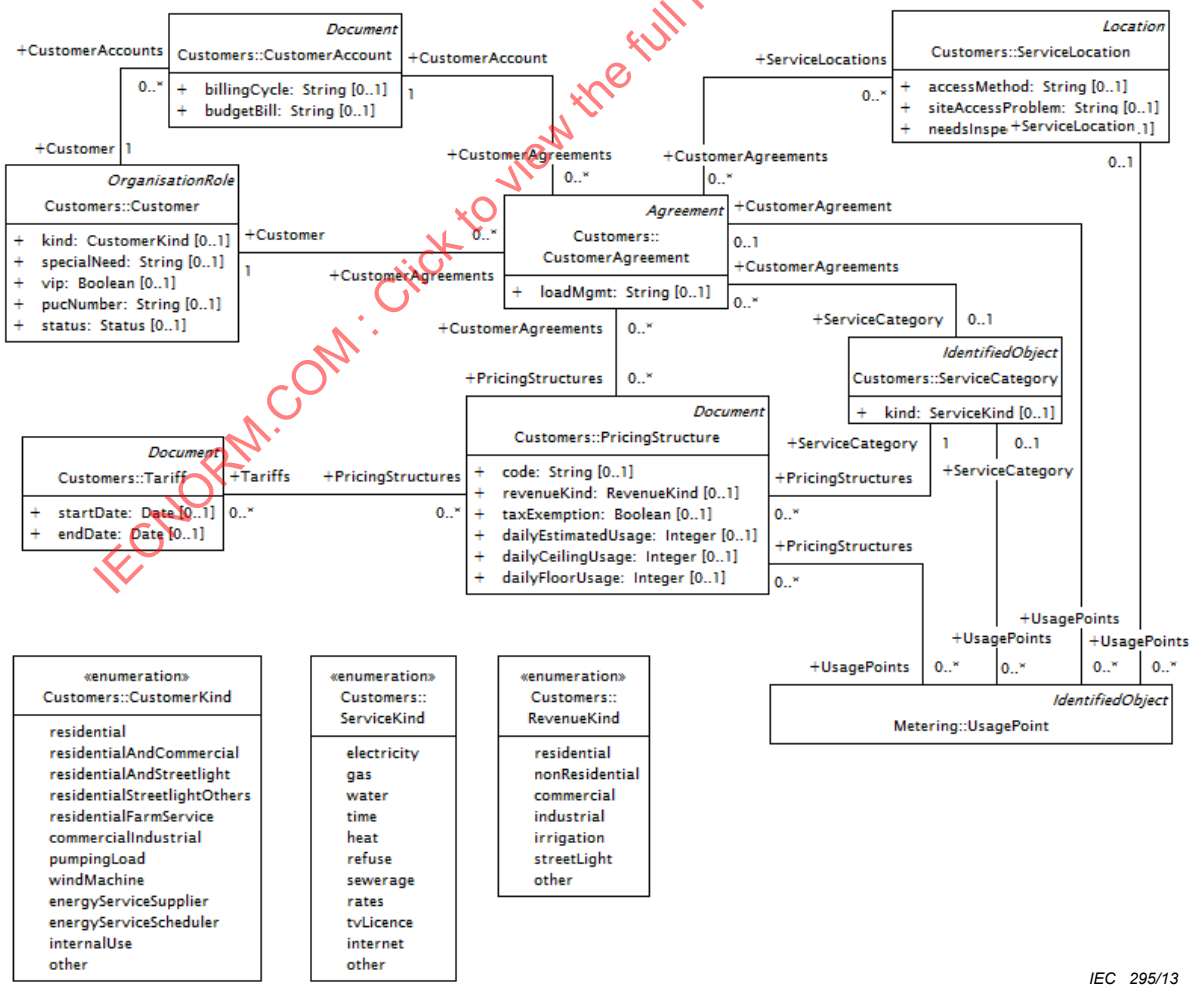


Figure 18 – Modèle Customers DCIM

Un **Customer** est une organisation qui reçoit les services d'un ou de plusieurs **ServiceSupplier**. Un **Customer** peut avoir plusieurs **CustomerAccount** et/ou **CustomerAgreement**.

Un **CustomerAccount** est le mécanisme de facturation et de paiement applicable au client. Il est utilisé pour créer les facturations (factures) à un **Customer** et pour en percevoir le règlement. Un **CustomerAccount** est généralement facturé selon un cycle défini de facturation. Un **CustomerAccount** peut prendre en charge la facturation globale de plusieurs **CustomerAgreement**. Il peut donc exister plusieurs **CustomerAgreement** relié à un **CustomerAccount** donné.

Un **CustomerAgreement** est l'accord conclu entre le **Customer** et le **ServiceSupplier** de rétribuer un service à un point de livraison de service spécifique (**UsagePoint**, voir modèle de comptage en 4.4.5.1). Il identifie certaines informations de facturation utilisées pendant la création de **Charge**, comprenant la **PricingStructure**.

Une **PricingStructure** est le regroupement des composants de tarification et des prix utilisés pour la création des **Charge** client et les critères d'admissibilité selon lesquels ces conditions peuvent être proposées à un client **Customer**. Les raisons qui justifient le regroupement comprennent la juridiction réglementaire, la classification client, les caractéristiques de site, la classification (c'est-à-dire le barème tarifaire, la structure de prix d'acompte, la structure de prix du service de distribution d'électricité, etc.) et les exigences en matière de comptabilité. Une **PricingStructure** peut englober un ou plusieurs **Tariff** qui sont souvent désignés barèmes tarifaires.

CustomerAccount, **PricingStructure** et **CustomerAgreement** sont tous des spécialisations de **Document**, établies directement ou par le biais de la classe **Agreement**. Cela permet de démontrer l'usage prévu de la classe **Document** en la spécialisant selon le cas et en n'utilisant jamais directement la classe **Document** pour les relations.

Une **ServiceCategory** est la catégorie (électricité, eau, gaz, etc.) du service fourni. Un **ServiceSupplier** est l'organisation qui fournit les services aux clients. Il existe plusieurs types de **ServiceSupplier**s (par exemple, entreprise de services publics, revendeur ou autre). Un **ServiceSupplier** donné peut fournir des services dans plusieurs instances de **ServiceCategory**.

Un **ServiceLocation** est un emplacement immobilier, communément appelé le local où un ou plusieurs services peuvent être fournis. Il peut avoir plusieurs **UsagePoint**; cependant, chaque **UsagePoint** ne concerne qu'une et une seule **ServiceCategory**. En complément, un **ServiceLocation** peut avoir plusieurs **UsagePoint** pour une **ServiceCategory** donnée.

NOTE La présente partie de la CEI 61968 (documentant DCIM11) fournit la modélisation client uniquement pour les besoins de la CEI 61968-9. Les prochaines éditions présenteront un modèle plus complet venant également à l'appui de la CEI 61968-8 et de la CEI 61968-6.

4.4.5 Modèle Metering

4.4.5.1 Généralités

Le paquetage **Metering** (c'est-à-dire comptage) introduit des classes qui sont nécessaires à l'intégration par les entreprises des systèmes de comptage et les informations et contrôles-commandes qu'elles échangent avec d'autres systèmes d'entreprises.

La Figure 19 illustre les classes spécifiques au domaine de comptage;[15] se référer également à la Figure 18 en 4.4.4 pour les classes associées aux clients, également applicables au domaine de comptage.

---

[15] Du fait du nombre de classes contenues, la figure ne montre que leurs noms et relations; 4.4.5.2 à 4.4.5.7 détaillent les sous-ensembles des classes associées.

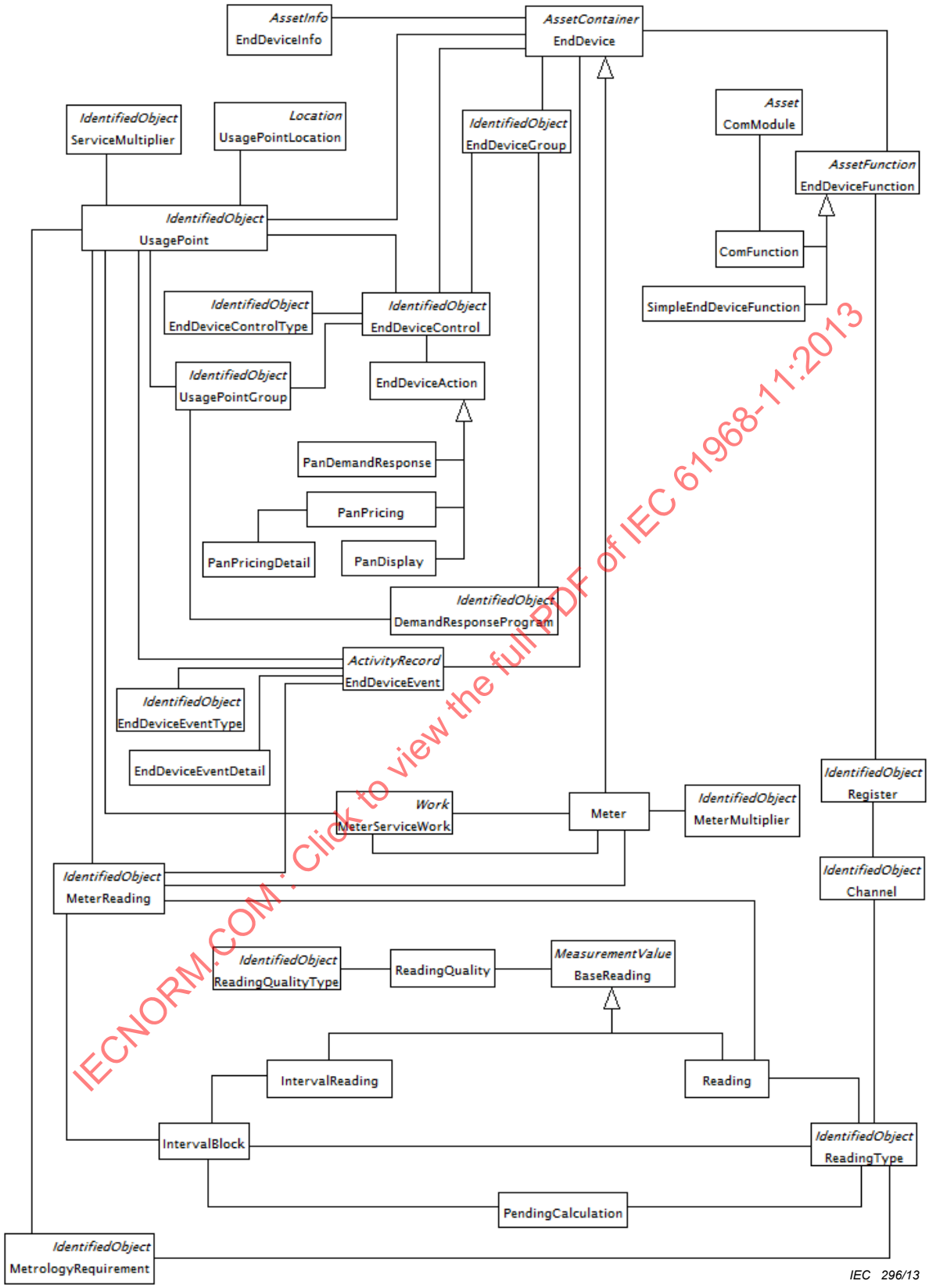


Figure 19 – Modèle de comptage du DCIM

Ces classes prennent en compte les entités types du domaine de comptage, ainsi que les concepts requis pour les échanges de données opérationnels. 4.4.5.2 à 4.4.5.7 décrivent de manière plus détaillée les sous-ensembles associés. 4.4.5.2 à 4.4.5.4 s'intéressent principalement aux entités de domaine qui nécessitent généralement d'être configurées pour

activer les échanges de données opérationnels. Ces types d'échanges de données opérationnels sont décrits de manière plus détaillée de 4.4.5.5 à 4.4.5.7.

4.4.5.2 Points d'usage

L'un des concepts de base du modèle de comptage est le **UsagePoint**, illustré avec ses relations à la Figure 20.

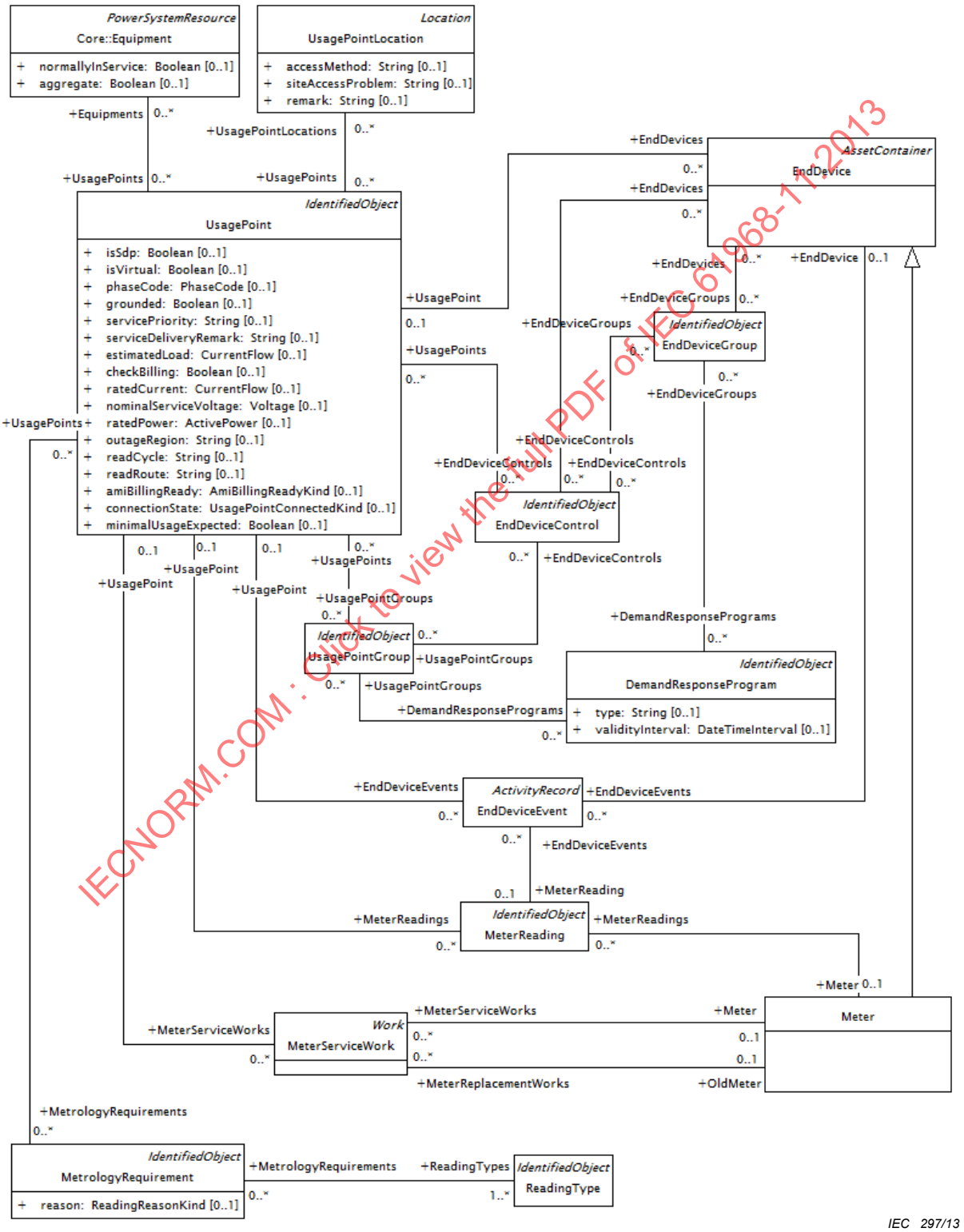


Figure 20 – Modèle de point d'usage DCIM

Le **UsagePoint** est le point logique ou physique sur le réseau auquel des **MeterReading** ou **EndDeviceEvent** peuvent être attribués. Il est utilisé à l'endroit où un **Meter** physique ou virtuel ou autre **EndDevice** peut être installé; cependant, la présence du dispositif n'est pas nécessaire.

Un **UsagePoint** peut être un point de livraison de service défini comme le point de démarcation sur un réseau où la propriété est transférée du **ServiceSupplier** au **Customer** (voir Figure 18).

Un **UsagePoint** peut avoir aucune ou plusieurs **MetrologyRequirement** qui définissent les **ReadingType** qui doivent être acquis ou calculés pour le **UsagePoint**.

Un **UsagePointLocation** définit l'emplacement physique des **UsagePoints**.

Les **MeterServiceWork** sont associés de manière dynamique à un **UsagePoint** et fournissent une spécification explicite dans le cas de travaux de remplacement de compteurs, par le biais de son extrémité d'association **OldMeter**.

UsagePoint est la classe établissant le lien entre le modèle orienté réseau des équipements connectés électriquement d'une part, et le modèle de biens et orienté locaux du domaine de comptage d'autre part. Ce lien peut être établi par la relation de **UsagePoint** à **Equipment**, la classe définie dans le paquetage de modèle **IEC61970::Wires** de la CEI 61970-301. Cette relation permet d'échanger la configuration des relations entre au moins les concepts suivants dans les deux domaines:

- Un **TransformerTank** fournit de l'énergie à un ou plusieurs **UsagePoints** connectés. Il est nécessaire que certaines fonctions de gestion des interruptions de service et de réseau de distribution connaissent cette relation.
- Un ou plusieurs **UsagePoint** peuvent représenter les **EnergyConsumer** cumulés tels que considérés par les systèmes types de salle de commande.
- les transformateurs de mesure (**CurrentTransformer** et **PotentialTransformer**) installés physiquement à un **UsagePoint** peuvent être modélisés de manière explicite par cette relation. Leurs informations de feuille de données peuvent être définies dans un système et partagées avec les autres systèmes.

Les **UsagePoint** peuvent être groupés de manière dynamique dans plusieurs **UsagePointGroup**, destinés à des opérations impliquant des **MeterReading**, **EndDeviceControl** et éventuellement d'autres opérations de comptage réalisées sur des groupes définis de **EndDevice** et/ou **UsagePoints**. Les messages entre les systèmes d'entreprise sont utilisés pour synchroniser chaque compréhension du système que se font les membres de chaque groupe.

4.4.5.3 Dispositifs finaux

Un autre concept essentiel du domaine de comptage modélisé dans le DCIM est celui des **EndDevice**, illustré avec ses relations à la Figure 21.

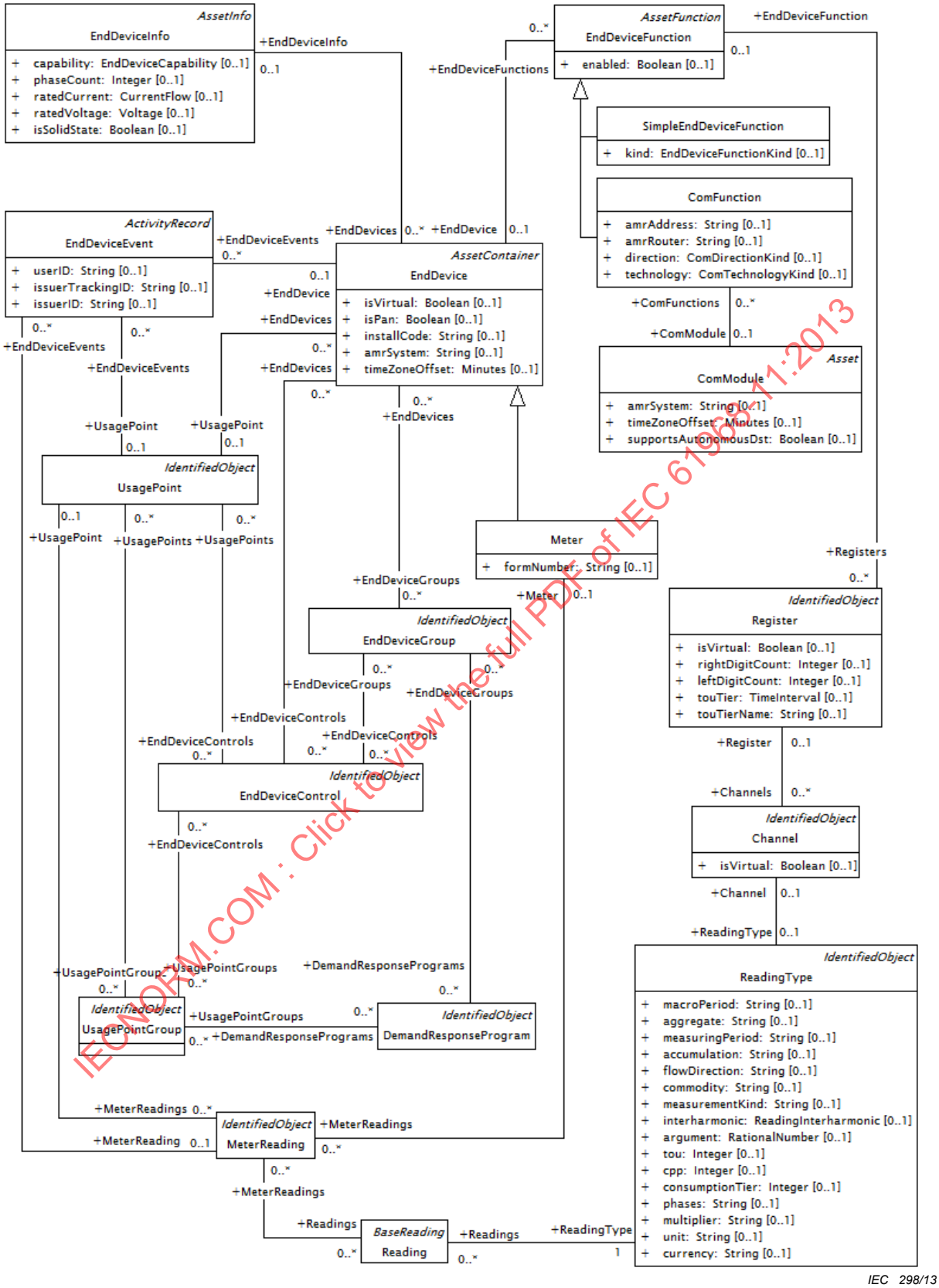


Figure 21 – Modèle de dispositif final DCIM

Un **EndDevice** est un **AssetContainer** qui réalise une ou plusieurs **EndDeviceFunction**s. Un type de **EndDevice** est représenté par un compteur **Meter** qui peut réaliser les fonctions de comptage, gestion des charges, branchement/débranchement, et autres fonctions. Un **EndDevice** peut également être utilisé pour modéliser un dispositif de réseau dans un local

(PAN – *premise area network*), qui peut réaliser des fonctions telles que l'affichage à domicile ainsi que la commande et la surveillance des équipements installés dans les locaux comme par exemple les appareils de climatisation, les réfrigérateurs, les pompes de piscine, etc. Un ou plusieurs dispositifs PAN peuvent être raccordés à un **Meter** ou à un **UsagePoint**.

Plusieurs **EndDevice**, et de ce fait également plusieurs **Meter**, peuvent être raccordés à un **UsagePoint** donné. Cependant, il est relativement courant de rencontrer des systèmes d'entreprise nécessitant un **UsagePoint** séparé pour chaque **Meter**.

Les **EndDevice** peuvent disposer d'une capacité de communication intégrée. De même, la capacité de communication des **EndDevice** peut être assurée par un ou plusieurs **ComModule**; Cette contenance(containment) est hérité(e) au moyen de l'association **AssetContainer–Asset**.

Un **Meter** peut être un **EndDevice** physique ou virtuel qui réalise la fonction de comptage. Un compteur physique existe lorsqu'un matériel du monde réel réalise la fonction de comptage. Un compteur virtuel est réalisé en l'absence de matériel du monde réel et peut par exemple être défini pour réaliser des fonctions telles que l'opération de cumul de la consommation d'au moins deux compteurs physiques.

Les **Meter**s peuvent avoir un ou plusieurs **Register**s, qui sont des dispositifs qui indiquent ou enregistrent les unités d'un produit de base ou autre grandeur mesurée. Chaque **Register** à tour de rôle peut avoir un ou plusieurs **Channel**. Chaque **Channel** est cependant réservé à un et un seul **ReadingType**.

Un **Channel** est un chemin simple permettant de collecter ou de consigner les valeurs d'un **Register** sur une durée donnée. Par exemple, un **Register** qui mesure la puissance active peut avoir deux **Channel**: l'un fournissant les relevés de comptage en quantité et l'autre fournissant des relevés d'intervalle d'une taille fixe.

Une instance de **ReadingType** est utilisée comme un identificateur unique qui spécifie les attributs requis pour caractériser pleinement un **Reading**. Un **Reading** est une valeur spécifique mesurée ou calculée par un **Meter** ou système.

NOTE Les attributs de **ReadingType**, actuellement définis comme des chaînes dans le modèle UML, détiennent les valeurs d'entier définies à l'Annexe C de la CEI 61968-9:2009 dans les échanges de données. Il est prévu de modéliser ces valeurs en libellés d'énumération dans certaines des prochaines éditions de la présente CEI 61968-11. L'harmonisation sera nécessaire avec les types d'énumération équivalents définis dans le paquetage de modèle **IEC61970::Domain** de la CEI 61970-301, pour les attributs suivants de **ReadingType**: **phases**, **multiplier**, **unit** et **currency**.

4.4.5.4 Événements de configuration

Contrairement aux modèles de réseau électrique dans lesquels il existe généralement un système qui représente la source de données maître, le domaine de comptage peut comprendre plusieurs systèmes d'enregistrement, chacun d'eux étant un maître pour uniquement un sous-ensemble d'entités métier. Par conséquent, le suivi des changements de configuration nécessite une haute flexibilité, il est donc modélisé avec les instances de **ConfigurationEvent**, comme cela est illustré à la Figure 22.

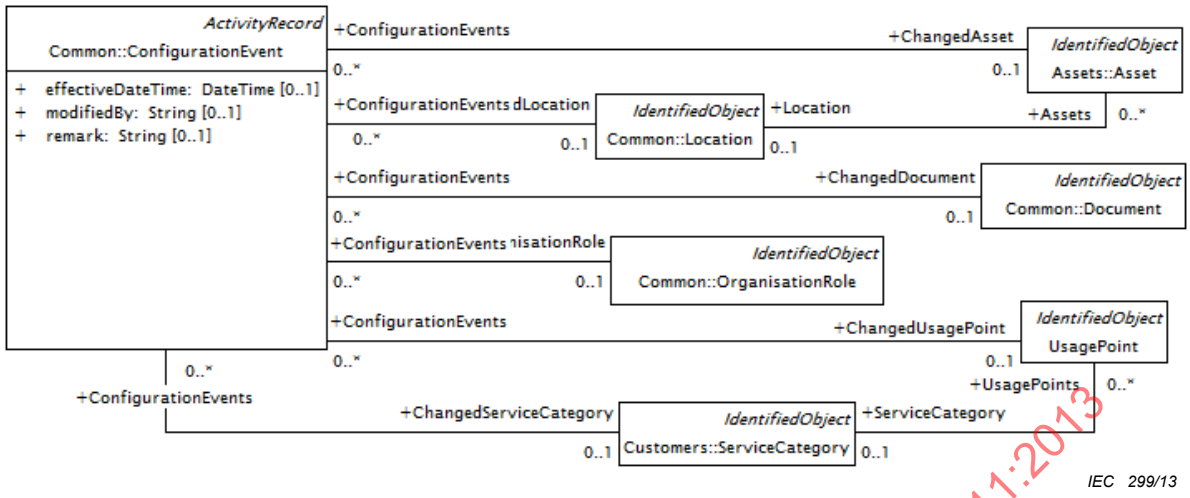


Figure 22 – Événements de configuration pour le comptage

Le **ConfigurationEvent** est une spécialisation de **ActivityRecord**, utilisé exclusivement pour l'échange de données relatives aux changements de configuration de divers objets: **Asset**, **Location**, **Document**, **OrganisationRole**, **UsagePoint** et instance de **ServiceCategory**.

4.4.5.5 Relevés de compteur

Les relevés de compteur sont échangés entre les systèmes d'entreprise pour prendre en charge des fonctions telles que la facturation, la réponse à demande, la gestion de charge, la planification du système et la protection des recettes. La Figure 23 illustre les classes qui prennent en charge cette sorte d'échange de données.

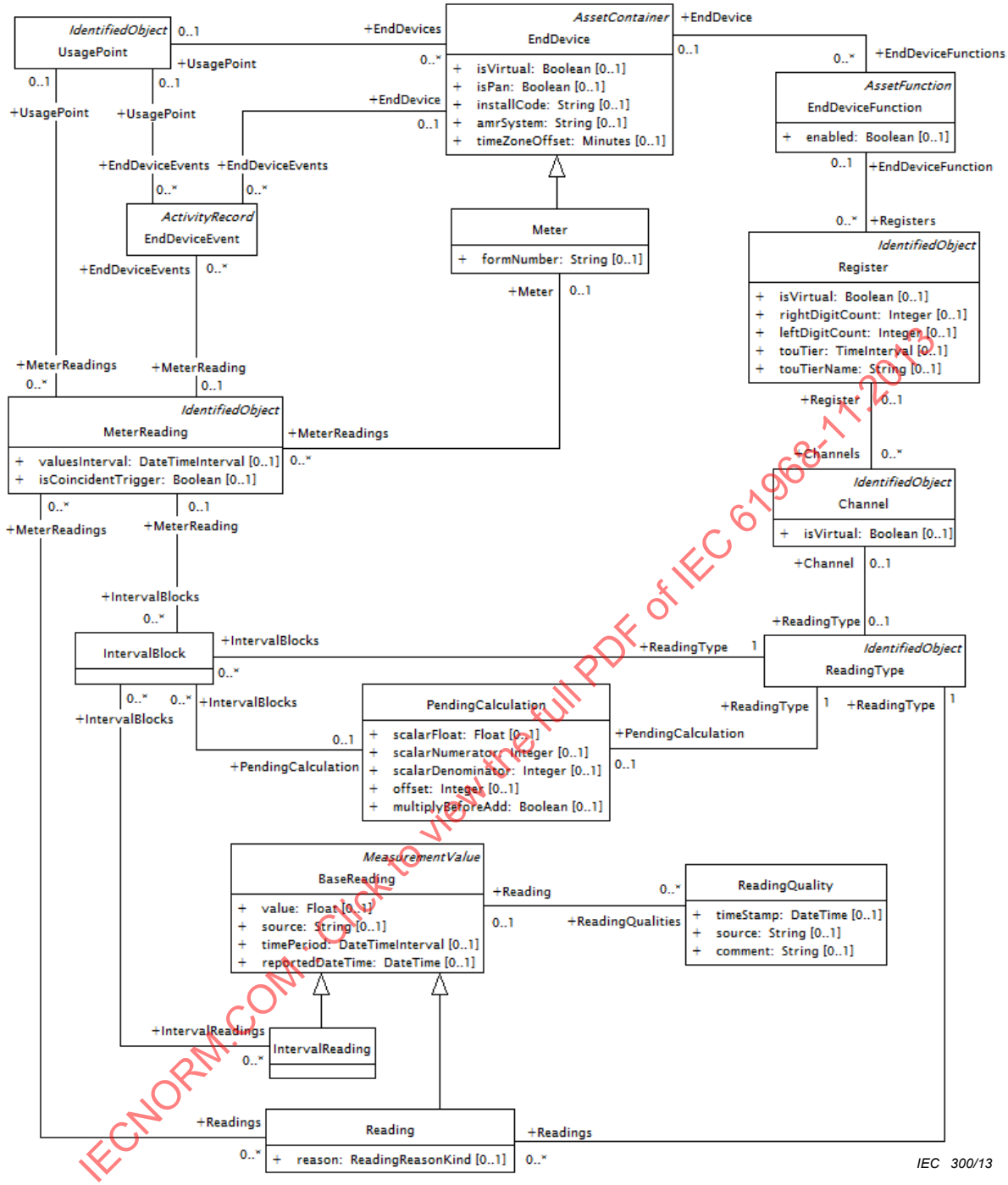


Figure 23 – Modèle de relevés de compteur DCIM

Les **MeterReading** peuvent comprendre diverses combinaisons de **Reading** et de **IntervalBlock**. Les **Reading** représentent des valeurs (mesurées ou calculées) à un moment spécifique dans le temps et qui sont caractérisées par un **ReadingType** et une ou plusieurs instances de **ReadingQuality**. Les **IntervalBlock** représentent des ensembles de **Reading** à des intervalles de temps égaux.

Un cas type d'utilisation des **Reading** est celui d'un système de comptage ou système de gestion de données de compteur fournissant un système de facturation avec relevé de registres à minuit. Les **IntervalBlock** peuvent être utilisés pour fournir un système d'informations clients avec des relevés de consommation plus détaillés, par exemple pendant des durées de 15 min ou d'1 h. Le dernier cas peut se révéler nécessaire directement pour la

facturation ou pour répartir la consommation selon des durées d'utilisation qui peuvent être facturées selon différents taux.

NOTE Bien que **Reading** et **IntervalReading** spécialisent la classe **MeasurementValue**, définie dans le paquetage de modèle **IEC61970::Meas** de la CEI 61970-301, il n'existe que très peu d'éléments communs entre les deux modèles pour les mesures. Le seul attribut issu de **MeasurementValue** utilisé est **timeStamp**; aucune de ses extrémités d'associations n'est utilisée. Même s'il est possible d'associer un **Reading** à une instance de **Measurement**, les définitions de source, valeur, qualité et type sont totalement différentes. Par conséquent, pour les applications ou systèmes qui souhaiteraient traiter un **Reading** comme une **MeasurementValue**, ou inversement, le DCIM normalisé ne fournit actuellement aucun support pour la transformation entre les deux représentations.

4.4.5.6 Commandes et événements de dispositif final

Certains dispositifs finaux sont capables de réaliser des actions en réponse à une commande ou de détecter l'occurrence d'événements particuliers. Ils peuvent également capturer et transmettre à d'autres systèmes des détails spécifiques sur ces événements. Les classes de modèle DCIM prenant en charge cette sorte d'échange de données sont illustrées à la Figure 24.

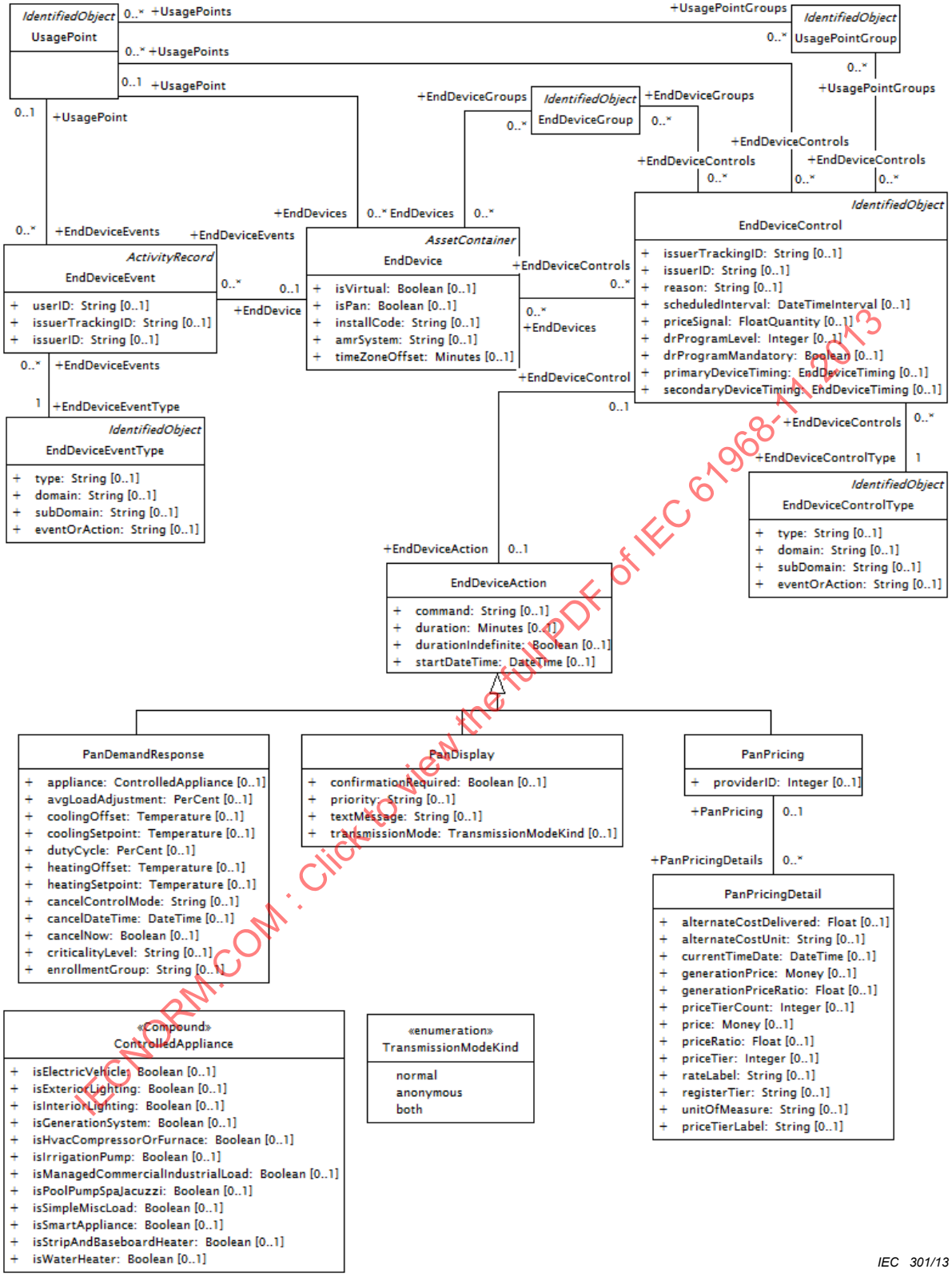


Figure 24 – Modèle de commandes et événements de dispositif final DCIM

Un **EndDevice** peut agir sur réception des commandes **EndDeviceControl**. Des exemples comprennent le branchement ou le débranchement automatisé d'un service par un **Meter**, la réinitialisation du registre de demande d'un **Meter**, l'affichage d'un message texte sur un dispositif PAN, la communication d'informations de tarification à un dispositif PAN ou autre **EndDevice**, etc.

Les commandes **EndDeviceControl** sont généralement demandées par des systèmes d'entreprise tels que le système d'informations clients, et sont en dernier lieu transmises au système de comptage pour exécution. Une **EndDeviceControl** spécifie l'action de commande souhaitée en utilisant un **EndDeviceControlType**. Le système d'entreprise peut demander une commande pour exécution immédiate ou exécution à des date et heure ultérieures.

Les systèmes peuvent cibler les **EndDeviceControl** aux **EndDevice** (ou **Meter**s) ou **UsagePoint** individuels. En variante, ils peuvent cibler un ou plusieurs **EndDeviceGroup** ou **UsagePointGroup**.

Certains **EndDevice** peuvent être programmés pour commander d'autres équipements qui ne représentent pas l'objet direct des **EndDeviceControl**. Par exemple, un dispositif PAN peut recevoir les informations détaillées d'un événement de réponse à demande (avec une **EndDeviceControl**) qui donneront finalement lieu à la mise sous ou hors tension des équipements tels que appareils de climatisation, pompes de piscine, etc. Pour cet échange de données, les informations requises pour "programmer" le dispositif PAN sont communiquées dans la **EndDeviceControl** avec une **EndDeviceAction** associée (qui est ultérieurement spécialisée en **PanDemandResponse**, **PanPricing** ou **PanDisplay**).

La Figure 24 montre également les classes applicables à la modélisation des **EndDeviceEvent**. Elles sont détectées et capturées par les **EndDevice** et transmises à d'autres systèmes d'entreprise. Les **EndDeviceEvent** proviennent fréquemment d'un système de comptage (en communication directe avec le **EndDevice**) et sont transmis à d'autres systèmes comme des événements déclencheurs pour l'exécution de diverses actions logiques métier. Des exemples de **EndDeviceEvent** comprennent la perte ou le rétablissement de la puissance, la perte ou le rétablissement des fonctions de communication, les alarmes de manipulations abusives de compteur, les indicateurs de rotation inverse et de nombreux autres éléments de ce type. Par exemple, les alarmes de perte de puissance peuvent être communiquées à un système de gestion des interruptions de service pour diagnostic de l'étendue de l'interruption du service. Ou encore, les alarmes de rotation inverse et de manipulations abusives peuvent être transmises à un système chargé de détecter le vol d'énergie ou autres problèmes de protection des recettes.

Un **EndDeviceEvent** spécifie l'événement avec un **EndDeviceEventType**.

Les événements détectés par les **EndDevice** peuvent se produire de manière spontanée (sans sollicitation) ou résulter directement d'une commande **EndDeviceControl**. Dans le dernier cas, ils sont souvent désignés "événements consécutifs". Les **EndDeviceEvent** sont généralement associés à un **UsagePoint**.

4.4.5.7 Demandes de service de compteur

Le modèle DCIM fournit également un support de base aux échanges de données dans le contexte de la gestion des travaux associée aux dispositifs finaux. Les classes correspondantes ont été illustrées à la Figure 20.

MeterServiceWork est une spécialisation de **Work**, où un **EndDevice** est impliqué.

Par exemple, il peut être nécessaire d'installer, de déplacer ou de configurer des **Meter** du fait de l'enregistrement ou inscription d'un nouveau **Customer**, du retrait d'un **Customer** ou du transfert d'un **Customer** d'un **ServiceSupplier** à un autre. Il peut également être nécessaire de remplacer un **Meter** ce qui implique de retirer l'ancien **Meter**, d'installer le nouveau **Meter** et de configurer le nouveau **Meter** selon les besoins du système de comptage.

MeterServiceWork prend également en charge la collecte des **MeterReading** réalisée conjointement au travail de service réalisé.

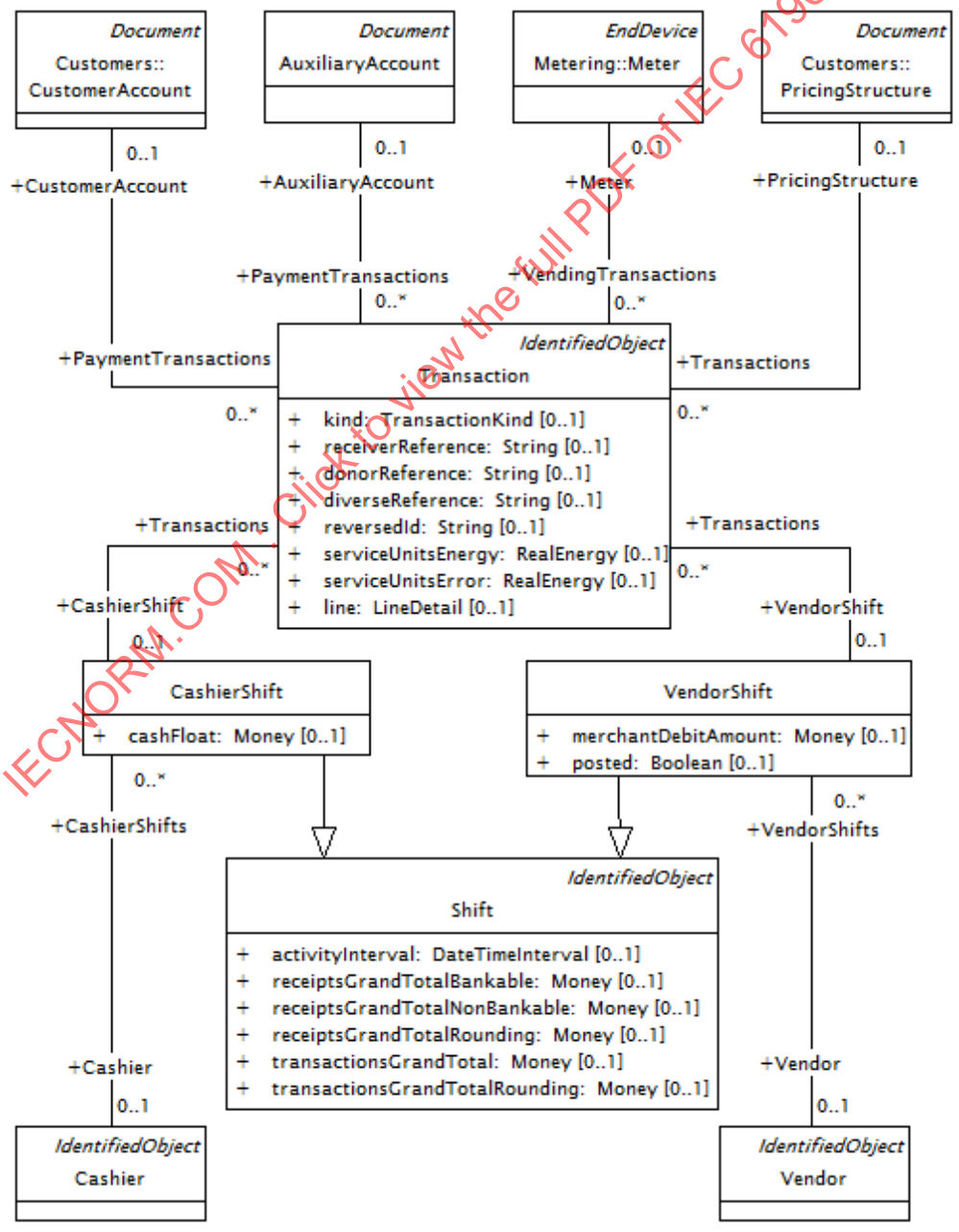
4.4.6 PaymentMetering

4.4.6.1 Généralités

Le Paymentmetering est une extension du paquetage de modèle **Metering**. Il contient les classes d'information qui prennent en charge les applications spécialisées telles que le comptageprépayé. Ces classes sont généralement associées à la collecte et au contrôle des recettes provenant du client en paiement d'un service fourni.

4.4.6.2 En transaction

Un système de paiement de comptage facilite généralement les transactions financières entre un client et un prestataire de service. Les informations pertinentes décrivant ces transactions sont typiquement enregistrées dans le système de paiement et ces informations sont ensuite échangées avec un autre système tel que le système de facturation ou d'informations clients. Un exemple type de la réalisation d'un tel plan d'enregistrement d'informations utilisant certaines des classes présentes dans le paquetage **PaymentMetering** est montré à la Figure 25.



IEC 302/13

Figure 25 – Modèle de transaction du DCIM

Le noyau d'informations dans ce modèle est la classe **Transaction**, qui capte toutes les informations pertinentes relatives à la transaction et aussi inclut des informations étendues telles que la date où le paiement est effectué par rapport à un **CustomerAccount**, le paiement par rapport à un **AuxiliaryAccount**, l'achat d'un **Token** prépayé pour un compteur de service de prépaiement et la tarification qui a été utilisée pour calculer la somme facturée pour cette vente.

Les informations relatives à la transaction peuvent en plus être rassemblées et triées en groupements **CashierShift** et **VendorShift** à des fins de comptabilité et de rapprochement opposables à un **Cashier** et à un **Vendor** qui sont comptables des recettes collectées au cours de la **Transaction** particulière.

4.4.6.3 Acquittement

Une transaction implique généralement un acquit de recettes de la part du client, qui peut se présenter sous la forme d'espèces, de chèque ou de carte par exemple. La saisie et l'échange consécutif d'informations décrivant les propriétés de ces recettes peuvent être réalisés au moyen de l'exemple de modèle montré à la Figure 26.

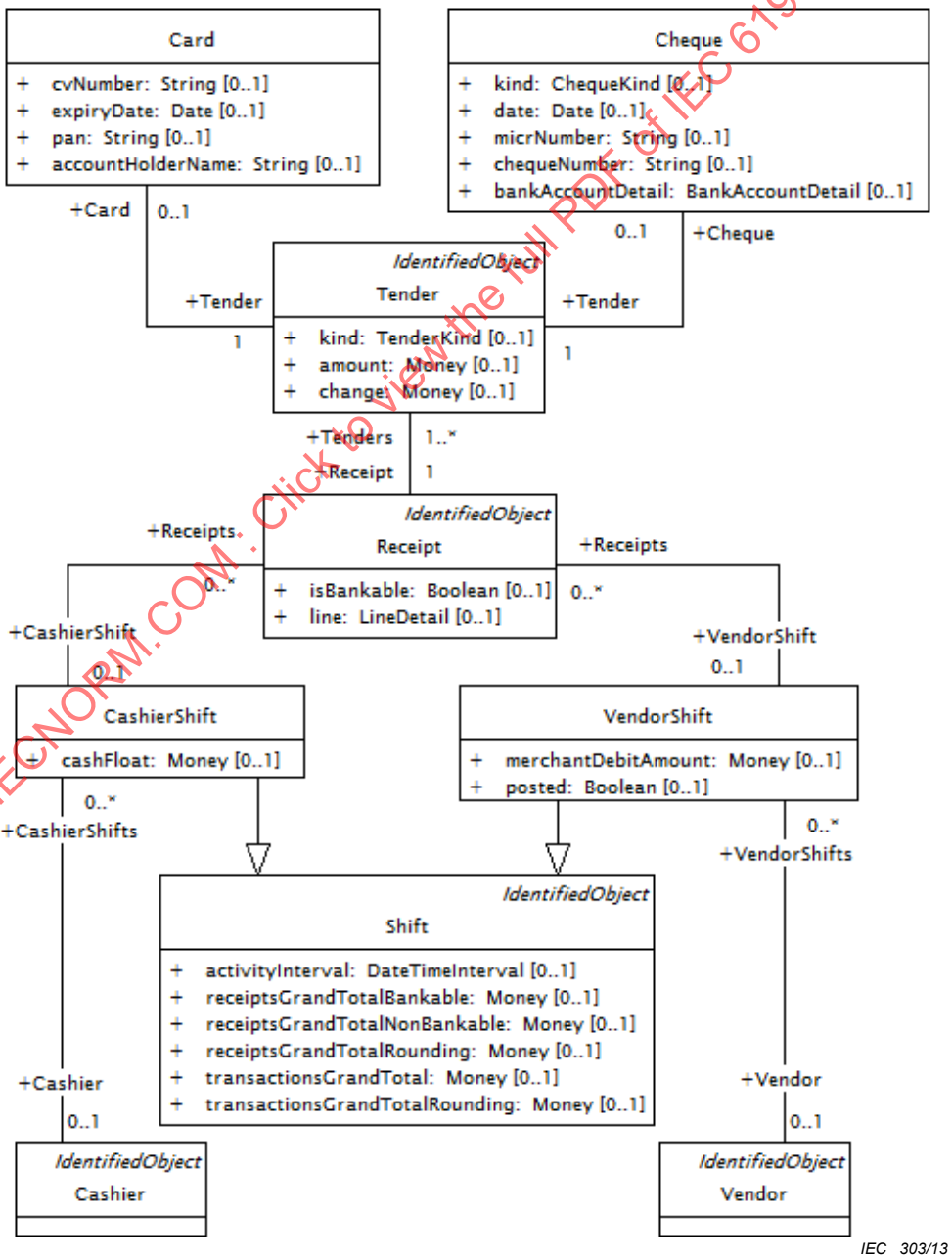


Figure 26 – Modèle d'acquittement du DCIM

Lorsqu'un client offre un paiement au cours d'une transaction, l'information est typiquement enregistrée dans les classes **Receipt**, **Tender**, **Card** et **Cheque**.

Les informations relatives à l'acquit peuvent en plus être rassemblées et triées en groupements **CashierShift** et **VendorShift** à des fins de comptabilité et de rapprochement opposables à un **Cashier** et à un **Vendor** qui sont comptables des recettes collectées au cours de la **Transaction** particulière.

4.4.6.4 Paiements auxiliaires

En plus des paiements types effectués par les clients pour les services fournis par le prestataire de service tel qu'un service public, il est souvent exigé d'acquitter les paiements pour d'autres éléments tels que créances, taxes, impôts, amendes municipales, redevances TV, taxes d'enlèvement des ordures, etc. La collecte de ces recettes peut être intégrée avec des ventes de jetons et des paiements de compte client au moyen d'accords auxiliaires et de comptes auxiliaires, dont un exemple est donné à la Figure 27.

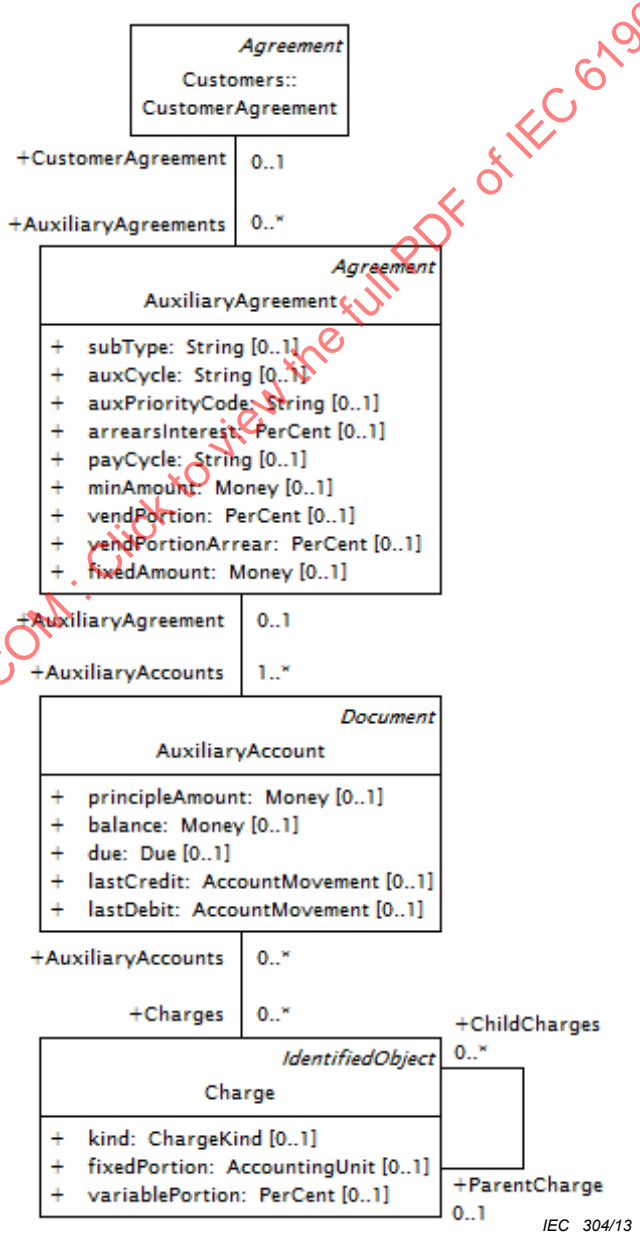


Figure 27 – Modèle d'accord auxiliaire du DCIM

AuxiliaryAgreement est essentiellement un prolongement de **CustomerAgreement** et capte les règles statiques indiquant comment le compte auxiliaire est géré. Il capte les informations dynamiques relatives aux charges et paiements effectués par le compte.

Charge permet des structures imbriquées de charges devant être imputées au compte **CustomerAccount** conformément aux règles établies dans **AuxiliaryAgreement** et prend en compte les coûts fixes, coûts variables et coûts en pourcentage.

4.4.6.5 Tarification et structures de tarification

Les structures de tarification peuvent contenir des tarifs dont la structure et le fonctionnement sont souvent assez complexes. La plupart des tarifs imposent des chargesqui sont basées sur la durée ou basées sur la consommation et, dans les deux cas, basées sur un intervalle. Un modèle pour réaliser ces structures de barèmes complexes est montré à la Figure 28.

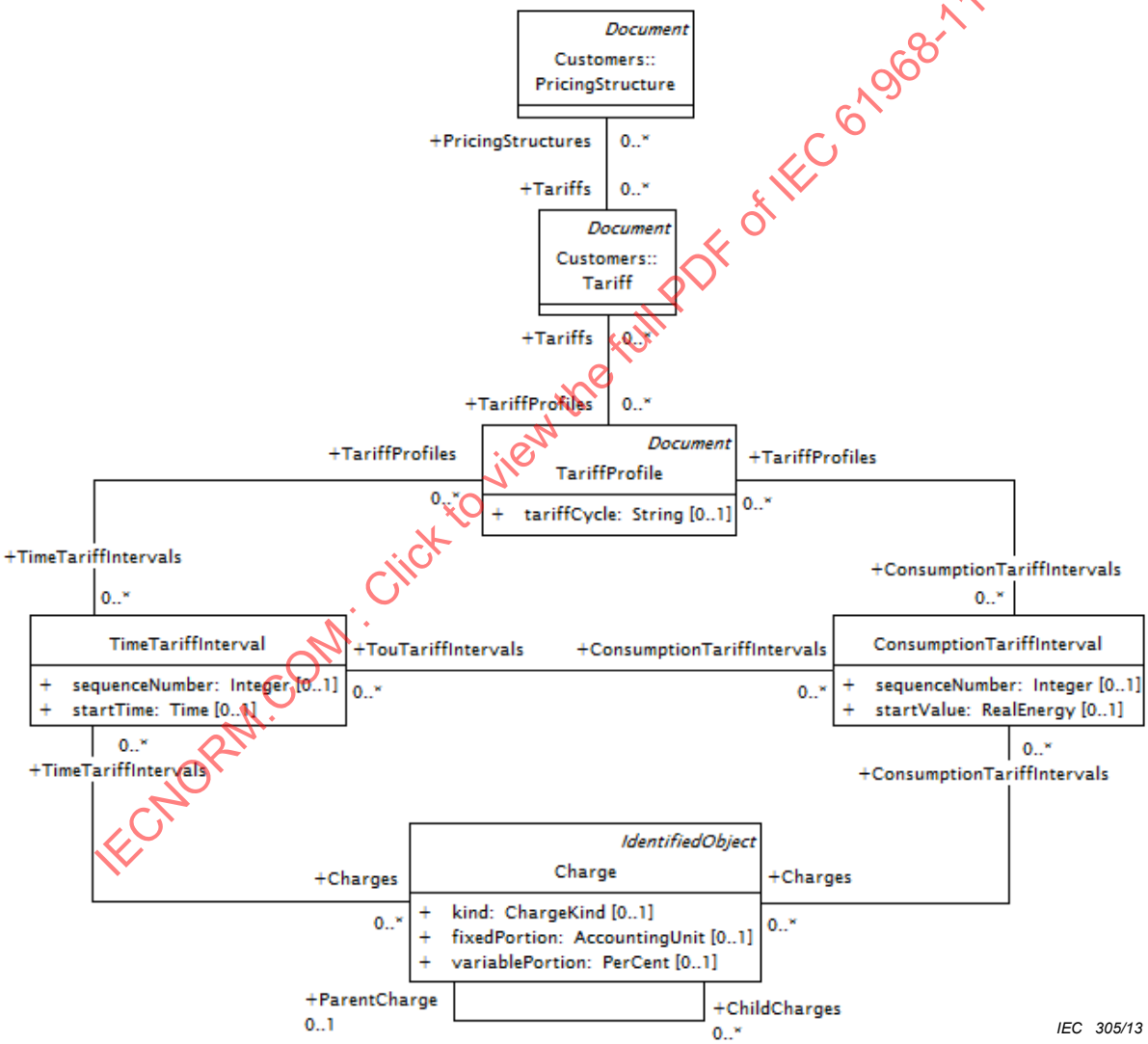


Figure 28 – Modèle de structure de tarification du DCIM

TariffProfile détermine le cycle de fonctionnement pour le **Tariff**, tel que horaire, quotidien, hebdomadaire, mensuel, etc. au bout duquel il se réinitialise pour démarrer de nouveau au début du processus.

TimeTariffInterval détermine l'heure de début d'un intervalle particulier et plusieurs instances de **TimeTariffInterval** peuvent être utilisées afin de construire une série d'intervalles de temps pour réaliser un tarif selon le temps d'utilisation par exemple.

En variante, **ConsumptionTimeInterval** détermine la valeur de départ d'un intervalle de consommation et plusieurs instances de **ConsumptionTimeInterval** peuvent être utilisées afin de construire une série d'intervalles de consommation pour réaliser un tarif à tranches ou un tarif à échelons par exemple.

Le prix par unité de service par intervalle de temps ou par intervalle de consommation est déterminé par la classe **Charge**, qui prend en charge les structures de coûts imbriquées et permet les coûts fixes, coûts variables et coûts en pourcentage.

Pour les structures de barèmes très complexes, il est permis de combiner **TimeTariffInterval** et **ConsumptionTimeInterval** pour prendre en charge simultanément les charges basées sur le temps et basées sur la consommation.

4.5 Autre

4.5 à 4.8 de la CEI 61970-301:— décrivent des outils de modélisation CIM, des extensions du CIM et des conventions de mise en œuvre.

5 Modèle détaillé

5.1 Vue générale

Le Modèle d'Information Commun (CIM) représente une vue logique complète des informations échangées parmi différents systèmes dans les services publics d'électricité. Cette définition inclut les classes et les attributs publics ainsi que leurs relations. Les paragraphes suivants décrivent la manière dont l'Article 6 est structuré. L'Article 6 est généré automatiquement par le modèle CIM maintenu avec les outils décrits en 4.6 de la CEI 61970-301:—.

5.2 Contexte

Le CIM est fractionné en sous-paquetages. Les classes au sein des paquetages sont répertoriées dans l'ordre dans lequel elles sont définies dans le modèle UML. Les attributs natifs de la classe sont d'abord répertoriés, suivis des attributs hérités dans l'ordre de la hiérarchie d'héritage. Les associations natives sont d'abord répertoriées pour chaque classe, suivies des associations héritées dans l'ordre de la hiérarchie d'héritage. Les associations sont décrites selon le rôle de chaque classe participant à l'association. Les extrémités d'associations sont répertoriées pour chaque classe à chaque extrémité d'une association.

La Figure 1 montre que le CIM de distribution (le présent document) dépend du CIM de base (la CEI 61970-301). Le présent document inclut la description détaillée du contenu du paquetage **IEC61968** seulement et fait référence à plusieurs classes, attributs et extrémités d'associations inclus dans le paquetage **IEC61970**.

Pour chaque paquetage, le modèle d'information de chaque classe est entièrement décrit. Les informations relatives aux attributs et extrémités d'association pour les attributs natifs et hérités sont documentées comme dans le Tableau 2 et dans le Tableau 3. Pour tous les éventuels attributs ou extrémités d'association hérités, la colonne "description" contiendra le texte indiquant que les attributs sont hérités d'une classe spécifique. La colonne "description" pour les attributs natifs et extrémités d'associations contient la description réelle.

Tableau 2 – Documentation d'attribut

nom	type	description
native1	Float	Un attribut natif virgule flottante de la classe est décrit ici.
native2	ActivePower	Documentation pour un autre attribut natif du type ActivePower.
name	String	hérité de: IdentifiedObject

Dans la documentation d'attribut (Tableau 2), dans certains cas, un attribut est une constante, auquel cas l'expression «(const)» est ajoutée dans la colonne nom du tableau d'attributs. Dans ces cas, l'attribut a normalement une valeur initiale qui est précédée du signal «égal» et accolé au nom d'attribut.

Tableau 3 – Documentation des extrémités d'associations

[mult à partir de]	[mult vers] nom	type	description
[0..1]	[0..*] PowerSystemResources	PowerSystemResource	Toutes les ressources du système électrique à cet emplacement.
[0..1]	[0..*] ConfigurationEvents	ConfigurationEvent	Tous les événements de configuration créés pour cet emplacement.
[0..*]	[0..1] CoordinateSystem	CoordinateSystem	Système de coordonnées utilisé pour décrire les points de position de cet emplacement.
[1..1]	[0..*] PositionPoints	PositionPoint	Séquence des points de position décrivant cet emplacement, exprimée dans le système de coordonnées 'Location.CoordinateSystem'.
[0..1]	[0..*] Assets	Asset	Tous les biens à cet emplacement.
[0..1]	[0..*] DiagramObjects	DiagramObject	hérité de: IdentifiedObject
[1..1]	[0..*] Names	Name	hérité de: IdentifiedObject

Dans la documentation des extrémités d'associations (Tableau 3), la première colonne décrit la multiplicité en cette extrémité de l'association (c'est-à-dire, comment cette classe participe à l'association). La deuxième colonne décrit l'autre extrémité d'association. Sa multiplicité (cardinalité) est insérée entre des crochets. Le nom de l'extrémité d'association est répertorié en texte en clair. La classe à l'autre extrémité de l'association est donnée dans la troisième colonne. Une multiplicité zéro indique une association facultative. Une multiplicité «*» indique que n'importe quel nombre est autorisé. Par exemple, une multiplicité [1..*] indique qu'une plage allant de 1 à n'importe quel nombre plus grand est autorisée.

Lorsqu'une classe est une énumération, le tableau des attributs est remplacé par la documentation enums comme dans le Tableau 4, le type de chaque enum dans l'énumération n'est pas défini. Il n'existe pas de libellés d'énumération hérités pour une classe d'énumération.

Tableau 4 – Documentation des enums

libellé	description
steel	
lead	
lock	
other	

6 Paquetage supérieur CEI 61968

6.1 Généralités

À l'heure actuelle, les parties normatives du modèle viennent à l'appui des besoins d'échange d'informations définis dans la CEI 61968-3, la CEI 61968-4, la CEI 61968-9 et la CEI 61968-13.

La Figure 29 montre le diagramme de paquetages IEC61968Dependencies.

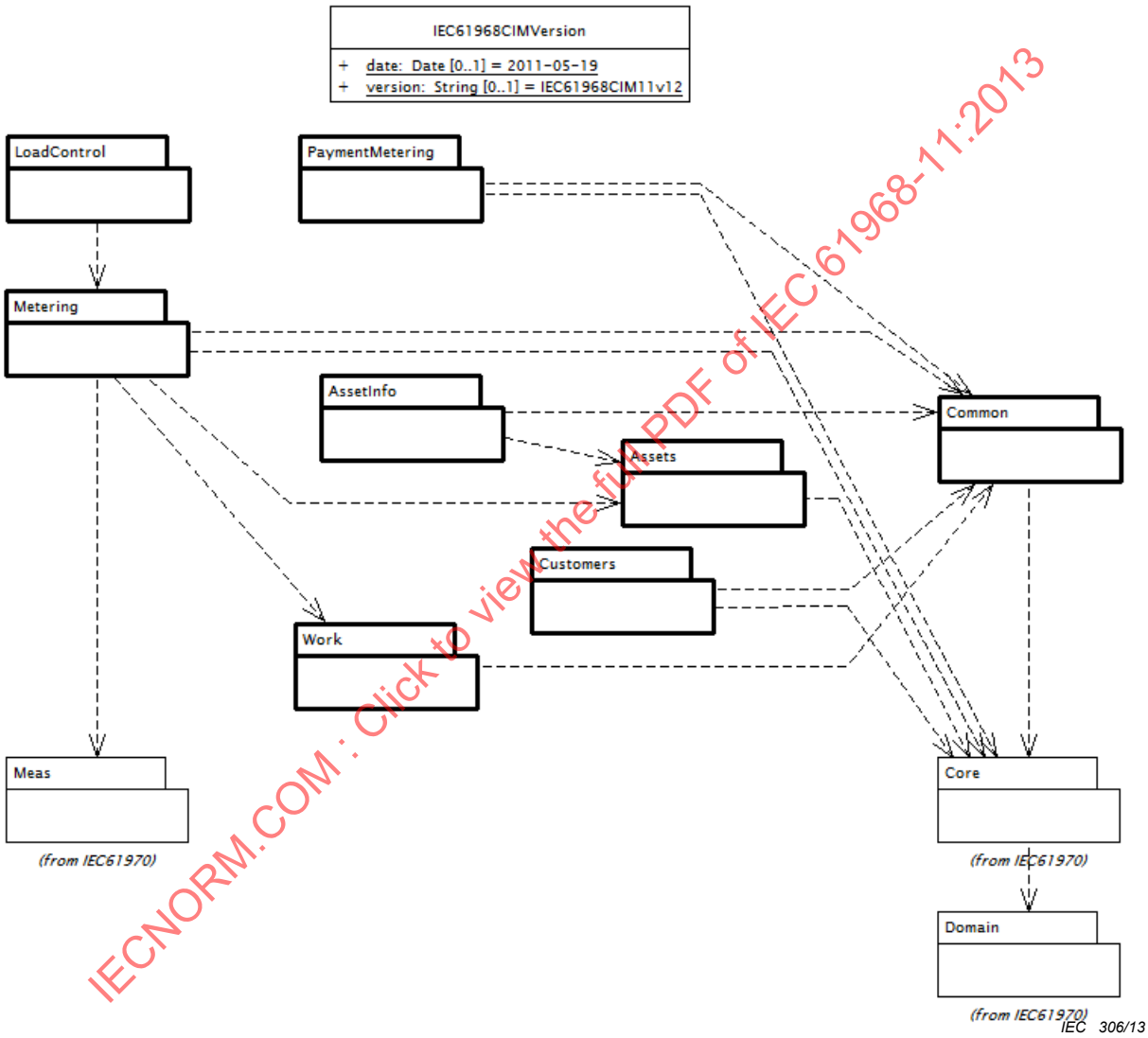


Figure 29 – Diagramme des paquetages IEC61968::IEC61968Dependencies

Ce diagramme montre la version et le contenu normatif des extensions du CIM pour la distribution ainsi que les dépendances entre les paquetages basées sur l'héritage uniquement..

Les paquetages indiqués en gras sont contenus et documentés à l'Article 6.

6.2 Classe racine IEC61968CIMVersion

Numéro de version de la CEI 61968 attribué à ce modèle UML.

Le Tableau 5 montre tous les attributs de IEC61968CIMVersion.

Tableau 5 – Attributs de IEC61968::IEC61968CIMVersion

nom	type	description
date=2011-08-10 (const)	Date	La forme est AAAA-MM-JJ par exemple le 5 janvier 2009 est 2009-01-05.
version=IEC61968CIM11v13 (const)	String	La forme est IEC61968CIMXXvYY où XX est la version principale du paquetage CIM et YY la version secondaire. Par exemple IEC61968CIM10v17.

6.3 Paquetage Common

6.3.1 Généralités

Ce paquetage contient les classes d'informations qui prennent en charge la gestion de la distribution en général.

La Figure 30 montre le diagramme de classe CommonInheritance.

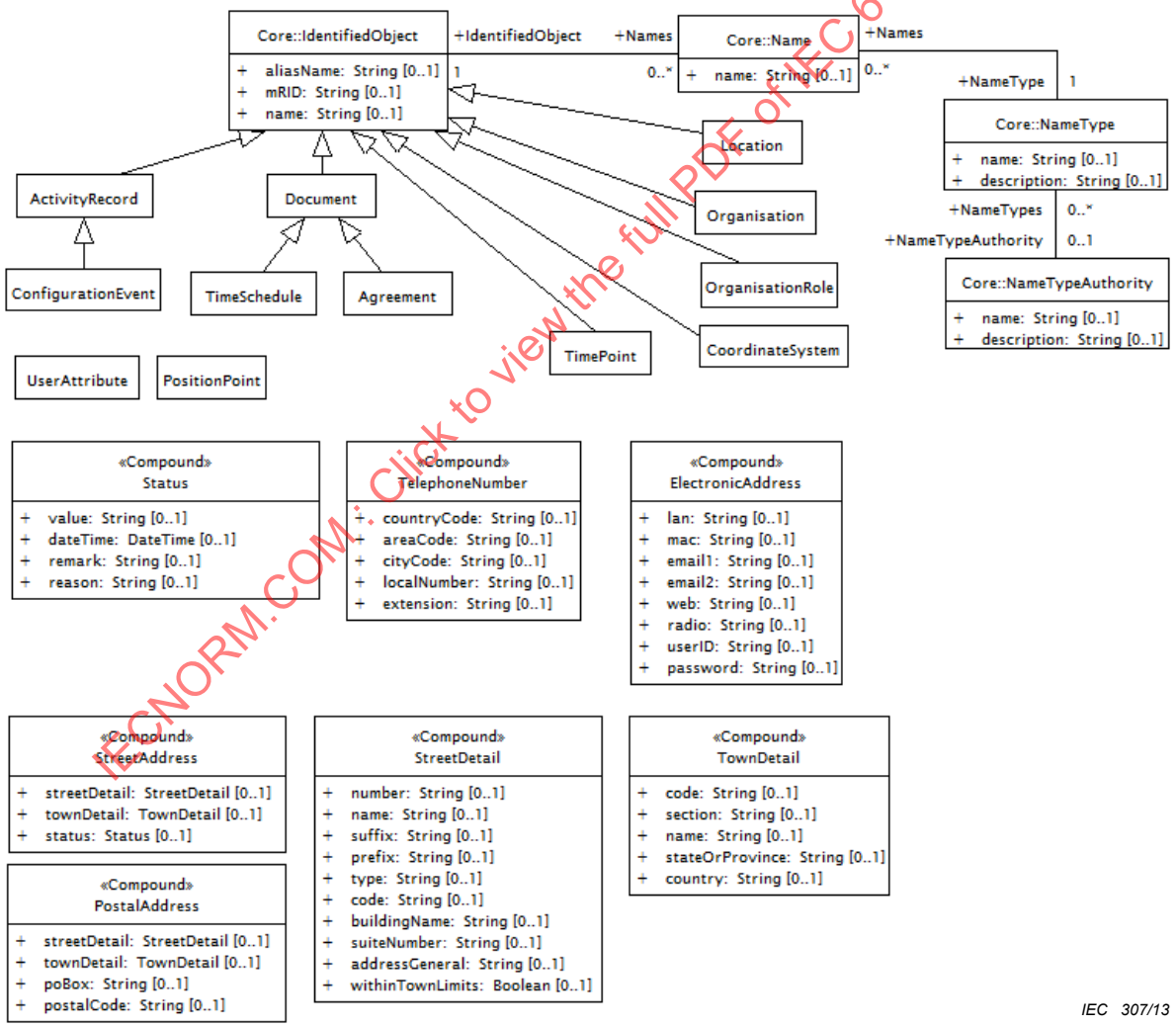


Figure 30 – Diagramme de classe Common::CommonInheritance

Ce diagramme montre la hiérarchie d'héritage pour les classes normatives issues de ce paquetage, ainsi que les énumérations et les types compound. Il montre également les classes applicables à la dénomination de IdentifiedObject.

La Figure 31 montre le diagramme de classe CommonOverview.

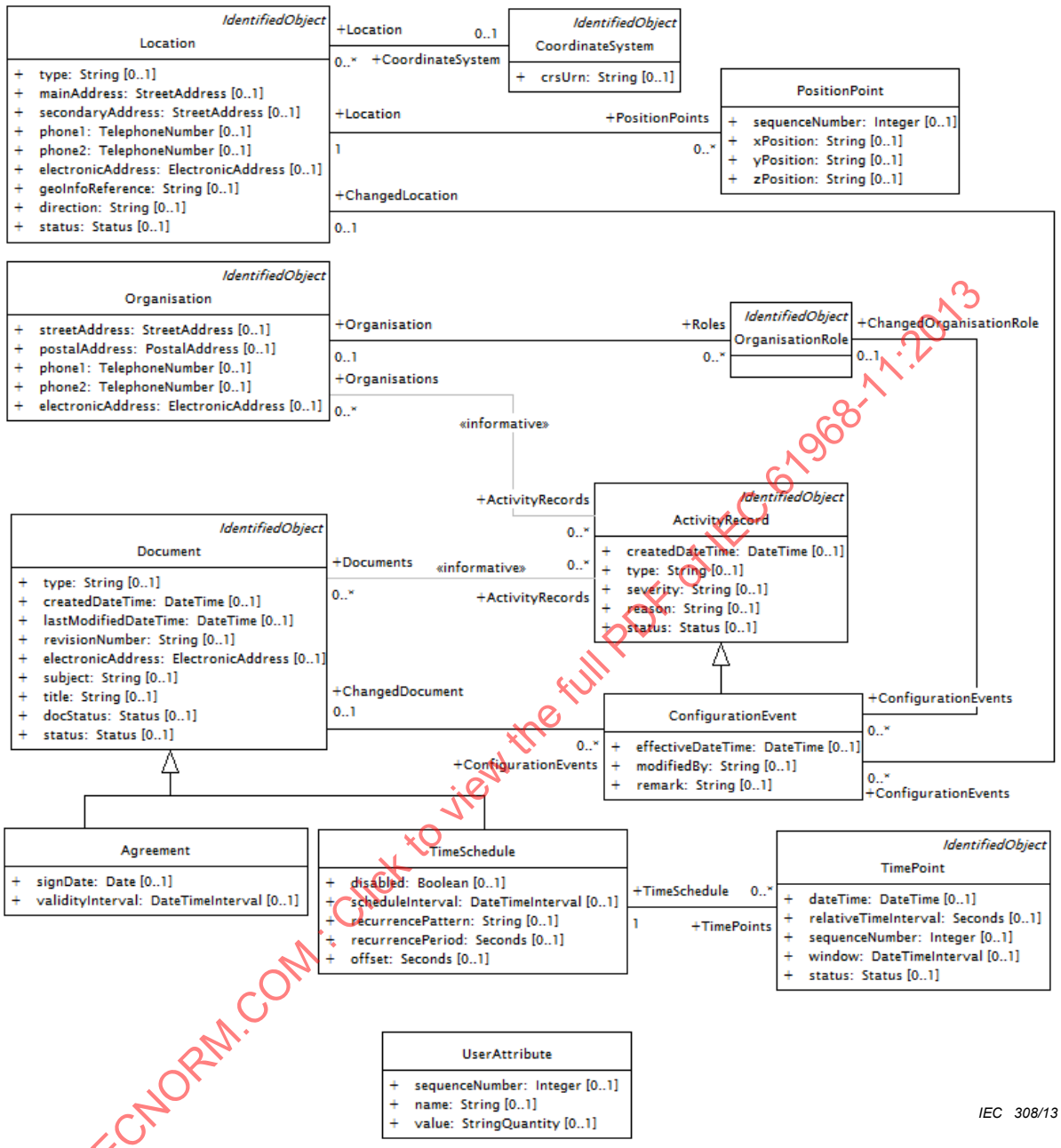


Figure 31 – Diagramme de classe Common::CommonOverview

Ce diagramme montre les classes normatives issues de ce paquetage.

6.3.2 Compound Status

Informations relatives au statut courant pertinentes pour une entité.

Le Tableau 6 montre tous les attributs de Status.

Tableau 6 – Attributs de Common::Status

nom	type	description
value	String	Valeur de Status à la date et heure «dateTime»; des changements d'état antérieurs ont pu avoir été conservés dans des instances de ActivityRecords associées à l'objet auquel ce Status s'applique.
dateTime	DateTime	Date et heure pour lesquelles «value» de status s'applique.
remark	String	Informations pertinentes relatives à la «value» (c'est-à-dire valeur) courante, sous forme de texte libre.
reason	String	Code de raison ou explication indiquant pourquoi un objet est passé à la «value» courante de status.

6.3.3 Compound PostalAddress

Informations d'usage général relatives à l'adresse postale.

Le Tableau 7 montre tous les attributs de PostalAddress.

Tableau 7 – Attributs de Common::PostalAddress

nom	type	description
streetDetail	StreetDetail	Détails relatifs à la rue.
townDetail	TownDetail	Détails relatifs à la ville.
poBox	String	Boîte postale.
postalCode	String	Code postal de l'adresse.

6.3.4 Compound StreetAddress

Informations d'usage général relatives à l'adresse de la rue.

Le Tableau 8 montre tous les attributs de StreetAddress.

Tableau 8 – Attributs de Common::StreetAddress

nom	type	description
streetDetail	StreetDetail	Détails relatifs à la rue.
townDetail	TownDetail	Détails relatifs à la ville.
status	Status	Statut de cette adresse.

6.3.5 Compound StreetDetail

Détails relatifs à la rue, dans le contexte de l'adresse.

Le Tableau 9 montre tous les attributs de StreetDetail.

Tableau 9 – Attributs de Common::StreetDetail.

nom	type	description
number	String	Appellation de l'emplacement spécifique dans la rue.
name	String	Nom de la rue.
suffix	String	Suffixe pour le nom de la rue. Par exemple: Nord, Sud, Est, Ouest.

nom	type	description
prefix	String	Préfixe pour le nom de la rue. Par exemple: Nord, Sud, Est, Ouest.
type	String	Type de rue. Les exemples comprennent: rue, rond-point, boulevard, avenue, route, ruelle, etc.
code	String	(si applicable) Les entreprises de services publics utilisent souvent des systèmes de référence externes, tels que ceux du système de cartographie du département d'urbanisation ou de l'arpenteur en chef, qui affectent des codes de référence globaux aux rues.
buildingName	String	(si applicable) Dans certains cas, l'emplacement physique du lieu d'intérêt n'a pas de point d'entrée direct à partir de la rue, mais peut être situé à l'intérieur d'une structure plus large telle qu'un bâtiment, un complexe, un bloc de bureaux, un appartement, etc.
suiteNumber	String	Numéro de l'appartement ou de la suite.
addressGeneral	String	Informations complémentaires relatives à l'adresse, par exemple une relai postal.
withinTownLimits	Boolean	True (c'est-à-dire vrai) si cette rue se situe dans les limites géographiques légales de la ville spécifiée (valeur par défaut).

6.3.6 Compound TownDetail

Détails relatifs à la ville, dans le contexte de l'adresse.

Le Tableau 10 montre tous les attributs de TownDetail.

Tableau 10 – Attributs de Common::TownDetail

nom	type	description
code	String	Code de la ville.
section	String	Section de la ville. Par exemple, il est courant d'avoir 36 sections par canton.
name	String	Nom de la ville.
stateOrProvince	String	Nom de l'état ou de la province.
country	String	Nom du pays.

6.3.7 Compound ElectronicAddress

Informations relatives à l'adresse électronique.

Le Tableau 11 montre tous les attributs de ElectronicAddress.

Tableau 11 – Attributs de Common::ElectronicAddress

nom	type	description
lan	String	Adresse sur le réseau local.
mac	String	adresse MAC (Media Access Control) .
Email1	String	Adresse de courriel (email) primaire.
Email2	String	Adresse de courriel (email) alternative.
web	String	Adresse web (Toile mondiale).
radio	String	Adresse radio.

nom	type	description
userID	String	Identificateur de l'utilisateur nécessaire pour se connecter (il peut s'agir de celui d'un individu, d'une organisation, d'un emplacement, etc.)
password	String	Mot de passe nécessaire pour se connecter.

6.3.8 Compound TelephoneNumber

Numéro de téléphone.

Le Tableau 12 montre tous les attributs de TelephoneNumber.

Tableau 12 – Attributs de Common::TelephoneNumber

nom	type	description
countryCode	String	Code de pays.
areaCode	String	Code de zone ou de région.
cityCode	String	(si applicable) Code de la ville.
localNumber	String	Partie principale (locale) de ce numéro de téléphone.
extension	String	(si applicable) Extension ou poste de ce numéro de téléphone.

6.3.9 ActivityRecord

Enregistre une activité pour une entité à un instant donné. L'activité peut concerner un événement déjà survenu ou une activité projetée.

Le Tableau 13 montre tous les attributs de ActivityRecord.

Tableau 13 – Attributs de Common::ActivityRecord

nom	type	description
createdDateTime	DateTime	Date et heure auxquelles cet enregistrement d'activité a été créé (différent de 'status.dateTime', qui est l'heure d'un changement de statut de l'objet associé, si applicable).
type	String	Type d'événement entraînant cet enregistrement d'activité.
severity	String	Niveau de gravité d'événement entraînant cet enregistrement d'activité.
reason	String	Raison de l'événement entraînant cet enregistrement d'activité, typiquement fournie lorsqu'il est déclenché par l'utilisateur.
status	Status	Informations sur la conséquence de l'événement entraînant cet enregistrement d'activité.
aliasName	String	hérité de: IdentifiedObject
mRID	String	hérité de: IdentifiedObject
name	String	hérité de: IdentifiedObject

Le Tableau 14 montre toutes les extrémités d'associations de ActivityRecord avec les autres classes.

Tableau 14 – Extrémités d'associations de Common::ActivityRecord avec les autres classes

[mult à partir de]	[mult vers] nom	type	description
[0..*]	[0..*] Assets	Asset	Tous les biens pour lesquels cet enregistrement d'activité a été créé.
[0.. 1]	[0..*] DiagramObjects	DiagramObject	hérité de: IdentifiedObject
[1..1]	[0..*] Names	Name	hérité de: IdentifiedObject

6.3.10 Agreement

Accord formel entre deux parties définissant les termes et conditions pour un ensemble de services. Les spécifications des services sont, à leur tour, définies via un ou plusieurs accords de service.

Le Tableau 15 montre tous les attributs de Agreement.

Tableau 15 – Attributs de Common::Agreement

nom	type	description
signDate	Date	Date à laquelle cet accord a été conclu entre les personnes et/ou organisations associées.
validityInterval	DateTimeInterval	Intervalle de date et de temps pendant lequel cet accord est valide (de son entrée en vigueur jusqu'à sa fin).
type	String	hérité de: Document
createdDateTime	DateTime	hérité de: Document
lastModifiedDateTime	DateTime	hérité de: Document
revisionNumber	String	hérité de: Document
ElectronicAddress	electronicAddress	hérité de: Document
subject	String	hérité de: Document
title	String	hérité de: Document
docStatus	Status	hérité de: Document
status	Status	hérité de: Document
aliasName	String	hérité de: IdentifiedObject
mRID	String	hérité de: IdentifiedObject
name	String	hérité de: IdentifiedObject

Le Tableau 16 montre toutes les extrémités d'associations de Agreement avec les autres classes.

Tableau 16 – Extrémités d'associations de Common::Agreement avec les autres classes

[mult à partir de]	[mult vers] nom	type	description
[0..1]	[0..*] ConfigurationEvents	ConfigurationEvent	hérité de: Document
[0..1]	[0..*] DiagramObjects	DiagramObject	hérité de: IdentifiedObject
[1..1]	[0..*] Names	Name	hérité de: IdentifiedObject

6.3.11 ConfigurationEvent

Utilisé pour consigner les détails sur la création, le changement ou la suppression d'une entité ou de sa configuration.

Le Tableau 17 montre tous les attributs de ConfigurationEvent.

Tableau 17 – Attributs de Common::ConfigurationEvent

nom	type	description
effectiveDateTime	DateTime	Date et heure auxquelles cet événement a été ou sera effectif.
modifiedBy	String	Origine/initiateur de la modification.
remark	String	Remarques en texte libre.
createdDateTime	DateTime	hérité de: ActivityRecord
type	String	hérité de: ActivityRecord
severity	String	hérité de: ActivityRecord
reason	String	hérité de: ActivityRecord
status	Status	hérité de: ActivityRecord
aliasName	String	hérité de: IdentifiedObject
mRID	String	hérité de: IdentifiedObject
name	String	hérité de: IdentifiedObject

Le Tableau 18 montre toutes les extrémités d'associations de ConfigurationEvent avec les autres classes.

Tableau 18 – Extrémités d'associations de Common::ConfigurationEvent avec les autres classes

[mult à partir de]	[mult vers] nom	type	description
[0..*]	[0..1] ChangedDocument	Document	Document dont le changement a donné lieu à cet événement de configuration.
[0..*]	[0..1] ChangedUsagePoint	UsagePoint	Point d'usage dont le changement a donné lieu à cet événement de configuration.
[0..*]	[0..1] ChangedAsset	Asset	Bien dont le changement a donné lieu à cet événement de configuration.
[0..*]	[0..1] ChangedServiceCategory	ServiceCategory	Catégorie de service dont le changement a donné lieu à cet événement de configuration.
[0..*]	[0..1] ChangedLocation	Location	Emplacement dont le changement a donné lieu à cet événement de configuration.
[0..*]	[0..1] ChangedOrganisationRole	OrganisationRole	Rôle d'organisation dont le changement a donné lieu à cet événement de configuration.
[0..*]	[0..*] Assets	Asset	hérité de: ActivityRecord
[0..1]	[0..*] DiagramObjects	DiagramObject	hérité de: IdentifiedObject
[1..1]	[0..*] Names	Name	hérité de: IdentifiedObject

6.3.12 CoordinateSystem

Système de référence de coordonnées.

Le Tableau 19 montre tous les attributs de CoordinateSystem.

Tableau 19 – Attributs de Common::CoordinateSystem

nom	type	description
crsUrn	String	Un Nom de ressource uniforme (URN – Uniform Resource Name) pour le système de référence de coordonnées (crs) utilisé pour définir les 'Location.PositionPoints'. Un exemple serait le code du European Petroleum Survey Group (EPSG) pour un système de référence de coordonnées, défini dans le URN dans le cadre de l'espace de noms (namespace) du Open Geospatial Consortium (OGC) comme: urn:ogc:def:uom:EPSG::XXXX, où XXXX est un code EPSG (une liste complète des codes peut être consultée sur le site web du EPSG Registry à l'adresse http://www.epsg-registry.org/). Pour définir le système de coordonnées comme étant WGS84 (latitude, longitude) avec un EPSG OGC, cet attribut serait urn:ogc:def:uom:EPSG::4236. Il convient qu'un profil limite ce code à un ensemble de URN admis, convenu par toutes les parties émettrices et réceptrices.
aliasName	String	hérité de: IdentifiedObject
mRID	String	hérité de: IdentifiedObject
name	String	hérité de: IdentifiedObject

Le Tableau 20 montre toutes les extrémités d'associations de CoordinateSystem avec les autres classes.

Tableau 20 – Extrémités d'associations de Common::CoordinateSystem avec les autres classes

[mult à partir de]	[mult vers] nom	type	description
[0..1]	[0..*] Location	Location	Tous les emplacements décrits par les points de position de ce système de coordonnées.
[0..1]	[0..*] DiagramObjects	DiagramObject	hérité de: IdentifiedObject
[1..1]	[0..*] Names	Name	hérité de: IdentifiedObject

6.3.13 Document

Classe parente pour les différents groupements d'informations recueillies et gérées en tant que partie intégrante d'un processus métier. Elle contiendra fréquemment des références à d'autres objets, tels que les biens, les personnes et les ressources du système électrique.

Le Tableau 21 montre tous les attributs de Document.

Tableau 21 – Attributs de Common::Document

nom	type	description
type	String	Catégorisation de ce document spécifique aux entreprises de service public, en fonction de leurs normes d'entreprise, de leurs pratiques et de leurs systèmes TI existants (par exemple, pour la gestion des biens, de la maintenance, des travaux, des interruptions de service, des clients, etc.)
createdDateTime	DateTime	Date et heure auxquelles ce document a été créé.

nom	type	description
lastModifiedDateTime	DateTime	Date et heure auxquelles ce document a été modifié la dernière fois. Les documents peuvent potentiellement être modifiés plusieurs fois au cours de leur durée de vie.
revisionNumber	String	Numéro de révision de ce document.
electronicAddress	ElectronicAddress	Adresse électronique.
subject	String	Sujet du document.
title	String	Titre du document.
docStatus	Status	Statut de ce document. Pour le statut du sujet que ce document représente (par exemple, Agreement, Work), utiliser l'attribut «status». Les exemples de valeurs pour «docStatus.status» sont projet, approuvé, annulé, etc.
status	Status	Statut du sujet (par exemple Agreement, Work) que ce document représente. Pour le statut du document lui-même, utiliser l'attribut «docStatus».
aliasName	String	hérité de: IdentifiedObject
mRID	String	hérité de: IdentifiedObject
name	String	hérité de: IdentifiedObject

Le Tableau 22 montre toutes les extrémités d'associations de Document avec les autres classes.

Tableau 22 – Extrémités d'associations de Common::Document avec les autres classes

[mult à partir de]	[mult vers] nom	type	description
[0..1]	[0..*] ConfigurationEvents	ConfigurationEvent	Tous les événements de configuration créés pour ce document.
[0..1]	[0..*] DiagramObjects	DiagramObject	hérité de: IdentifiedObject
[1..1]	[0..*] Names	Name	hérité de: IdentifiedObject

6.3.14 Location

L'endroit, le lieu ou le point de quelque chose où s'est trouvé, se trouve et/ou se trouvera quelqu'un ou quelque chose à un instant donné. Il est défini avec un ou plusieurs points de position (coordonnées) dans un système de coordonnées donné.

Le Tableau 23 montre tous les attributs de Location.

Tableau 23 – Attributs de Common::Location

nom	type	description
type	String	Catégorie par normes et pratiques professionnelles de l'entreprise de service public, relative à l'emplacement lui-même (par exemple, géographique, comptabilité par fonctions, etc., pas une propriété donnée qui existerait en cet emplacement).
mainAddress	StreetAddress	Adresse principale de l'emplacement.
secondaryAddress	StreetAddress	Adresse secondaire de l'emplacement. Par exemple, l'adresse de boîte postale peut avoir un code postal différent de celui indiqué dans «mainAddress».
phone1	TelephoneNumber	Numéro de téléphone.
phone2	TelephoneNumber	Numéro de téléphone supplémentaire.

nom	type	description
electronicAddress	ElectronicAddress	Adresse électronique.
geoInfoReference	String	(si applicable) Référence à une source d'informations géographiques, souvent extérieure à l'entreprise de service public.
direction	String	(si applicable) Direction qui permet aux équipes de terrain de trouver rapidement un bien donné. Pour un emplacement donné, tel qu'une adresse de rue, il s'agit de la direction relative dans laquelle trouver le bien. Par exemple, un réverbère peut se situer au coin «NW» (Nord-ouest) du site du client ou bien un «usagepoint» peut être situé au deuxième étage d'un immeuble d'appartements.
status	Status	Statut de cet emplacement.
aliasName	String	hérité de: IdentifiedObject
mRID	String	hérité de: IdentifiedObject
name	String	hérité de: IdentifiedObject

Le Tableau 24 montre toutes les extrémités d'association de Location avec les autres classes.

Tableau 24 – Extrémités d'associations de Common::Location avec les autres classes

[mult à partir de]	[mult vers] nom	type	description
[0..1]	[0..*] PowerSystemResources	PowerSystemResource	Toutes les ressources du système électrique en cet emplacement.
[0..1]	[0..*] ConfigurationEvents	ConfigurationEvent	Tous les événements de configuration créés pour cet emplacement.
[0..*]	[0..1] CoordinateSystem	CoordinateSystem	Système de coordonnées utilisé pour décrire les points de position de cet emplacement.
[1..1]	[0..*] PositionPoints	PositionPoint	Séquence des points de position décrivant cet emplacement, exprimée dans le système de coordonnées 'Location.CoordinateSystem'.
[0..1]	[0..*] Assets	Asset	Tous les biens à cet emplacement.
[0..1]	[0..*] DiagramObjects	DiagramObject	hérité de: IdentifiedObject
[1..1]	[0..*] Names	Name	hérité de: IdentifiedObject

6.3.15 Organisation

Organisation susceptible d'avoir des rôles comme entreprise de service public, entrepreneur, fournisseur, producteur, client, etc.

Le Tableau 25 montre tous les attributs de Organisation.

Tableau 25 – Attributs de Common::Organisation

nom	type	description
streetAddress	StreetAddress	Adresse de rue.
postalAddress	PostalAddress	Adresse postale, potentiellement différente de «streetAddress» (par exemple, une autre ville).
phone1	TelephoneNumber	Numéro de téléphone.
phone2	TelephoneNumber	Numéro de téléphone supplémentaire.

nom	type	description
electronicAddress	ElectronicAddress	Adresse électronique.
aliasName	String	hérité de: IdentifiedObject
mRID	String	hérité de: IdentifiedObject
name	String	hérité de: IdentifiedObject

Le Tableau 26 montre toutes les extrémités d'associations de Organisation avec les autres classes.

Tableau 26 – Extrémités d'associations de Common::Organisation avec les autres classes

[mult à partir de]	[mult vers] nom	type	description
[0..1]	[0..*] Roles	OrganisationRole	Tous les rôles de cette organisation.
[0..1]	[0..*] DiagramObjects	DiagramObject	hérité de: IdentifiedObject
[1..1]	[0..*] Names	Name	hérité de: IdentifiedObject

6.3.16 OrganisationRole

Identifie un moyen permettant à une organisation de participer aux activités d'une entreprise de services publics (par exemple, client, producteur, etc.).

Le Tableau 27 montre tous les attributs de OrganisationRole.

Tableau 27 – Attributs de Common::OrganisationRole

nom	type	description
aliasName	String	hérité de: IdentifiedObject
mRID	String	hérité de: IdentifiedObject
name	String	hérité de: IdentifiedObject

Le Tableau 28 montre toutes les extrémités d'associations de OrganisationRole avec les autres classes.

Tableau 28 – Extrémités d'associations de Common::OrganisationRole avec les autres classes

[mult à partir de]	[mult vers] nom	type	description
[0..1]	[0..*] ConfigurationEvents	ConfigurationEvent	Tous les événements de configuration créés pour ce rôle d'organisation.
[0..*]	[0..1] Organisation	Organisation	Organisation ayant ce rôle.
[0..1]	[0..*] DiagramObjects	DiagramObject	hérité de: IdentifiedObject
[1..1]	[0..*] Names	Name	hérité de: IdentifiedObject

6.3.17 Classe racine PositionPoint

Jeu de coordonnées spatiales qui déterminent un point défini dans le système de coordonnées spécifié dans 'Location.CoordinateSystem'. Utiliser une seule instance de point de position pour décrire un emplacement ponctuel. Utiliser une séquence de points de position pour décrire un objet orienté ligne (emplacement physique d'objets non ponctuels tels que câbles ou lignes), ou zone d'un objet (comme un poste ou une zone géographique – dans ce cas, ils ont les premier et dernier points de position avec les mêmes valeurs).

Le Tableau 29 montre tous les attributs de PositionPoint.

Tableau 29 – Attributs de Common::PositionPoint

nom	type	description
sequenceNumber	Integer	Numéro de séquence par rapport à zéro de ce point dans une série de points.
xPosition	String	Position sur l'axe X.
yPosition	String	Position sur l'axe Y.
zPosition	String	(si applicable) Position sur l'axe Z.

Le Tableau 30 montre toutes les extrémités d'associations de PositionPoint avec les autres classes.

Tableau 30 – Extrémités d'associations de Common::PositionPoint avec les autres classes

[mult à partir de]	[mult vers] nom	type	description
[0..*]	[1..1] Location	Location	Emplacement décrit par ce PositionPoint.

6.3.18 TimePoint

Instant dans une séquence d'instants relatifs à un TimeSchedule (c'est-à-dire à un planning).

Le Tableau 31 montre tous les attributs de TimePoint.

Tableau 31 – Attributs de Common::TimePoint

nom	type	description
dateTime	DateTime	Date et heure absolues pour cet instant. Pour les instants d'un calendrier, elles sont typiquement saisies manuellement, alors que pour l'instant basé sur un intervalle ou basé sur une séquence, elles sont dérivées.
relativeTimeInterval	Seconds	(si basé sur un intervalle) Instant relatif à l'heure de début programmée dans 'TimeSchedule.scheduleInterval.start'.
sequenceNumber	Integer	(si basé sur une séquence) Numéro de séquence relatif pour cet instant.
window	DateTimeInterval	Intervalle définissant la fenêtre de temps où cet instant est valide (par exemple: saisonnier, seulement les fins de semaine, pas pendant les fins de semaine, seulement de 8:00 à 5:00, etc.).
status	Status	Statut de cet instant.
aliasName	String	hérité de: IdentifiedObject
mRID	String	hérité de: IdentifiedObject
name	String	hérité de: IdentifiedObject

Le Tableau 32 montre toutes les extrémités d'associations de TimePoint avec les autres classes.

Tableau 32 – Extrémités d'associations de Common::TimePoint avec les autres classes

[mult à partir de]	[mult vers] nom	type	description
[0..*]	[1..1] TimeSchedule	TimeSchedule	Planning auquel cet instant appartient.
[0..1]	[0..*] DiagramObjects	DiagramObject	hérité de: IdentifiedObject
[1..1]	[0..*] Names	Name	hérité de: IdentifiedObject

6.3.19 TimeSchedule

Description de tout ce qui change avec le temps. Le TimeSchedule (planning) est utilisé pour accomplir une fonction simplement valuée du temps. Utiliser l'attribut hérité «type» pour donner des informations complémentaires sur ce calendrier, telles que périodicité (horaire, quotidien, hebdomadaire, mensuel, etc.), jour du mois, par date, calendrier (heures et dates spécifiques).

Le Tableau 33 montre tous les attributs de TimeSchedule.

Tableau 33 – Attributs de Common::TimeSchedule

nom	type	description
disabled	Boolean	True (c'est-à-dire vrai) si ce programme est désactivé.
scheduleInterval	DateTimeInterval	Intervalle de date et de temps du programme.
recurrencePattern	String	Intervalle auquel l'action programmée se répète (par exemple, premier mardi de chaque mois, dernier jour du mois, etc.).
recurrencePeriod	Seconds	Durée entre les instants, du début d'une période au début de la période suivante. Remarquer qu'un dispositif comme un compteur peut avoir plusieurs périodes d'intervalle (par exemple, 1 min, 5 min, 15 min, 30 min, ou 60 min).
offset	Seconds	Le décalage par rapport à minuit (à savoir, 0 h, 0 min, 0 s) pour que les instants périodiques commencent. Par exemple, pour un compteur à intervalles qui est réglé pour des intervalles de cinq minutes ('recurrencePeriod'=300=5 min), le réglage 'offset'=120=2 min entraînerait l'exécution d'événements programmés du compteur à chaque heure passée de 2 min, 7 min, 12 min, 17 min, 22 min, 27 min, 32 min, 37 min, 42 min, 47 min, 52 min, et 57 min.
type	String	hérité de: Document
createdDateTime	DateTime	hérité de: Document
lastModifiedDateTime	DateTime	hérité de: Document
revisionNumber	String	hérité de: Document
electronicAddress	ElectronicAddress	hérité de: Document
subject	String	hérité de: Document
title	String	hérité de: Document
docStatus	Status	hérité de: Document
status	Status	hérité de: Document
aliasName	String	hérité de: IdentifiedObject
mRID	String	hérité de: IdentifiedObject
name	String	hérité de: IdentifiedObject

Le Tableau 34 montre toutes les extrémités d'associations de TimeSchedule avec les autres classes.

Tableau 34 – Extrémités d'associations de Common::TimeSchedule avec les autres classes

[mult à partir de]	[mult vers] nom	type	description
[1..1]	[0..*] TimePoints	TimePoint	Séquence d'instants appartenant à ce planning.
[0..1]	[0..*] ConfigurationEvents	ConfigurationEvent	hérité de: Document
[0..1]	[0..*] DiagramObjects	DiagramObject	hérité de: IdentifiedObject
[1..1]	[0..*] Names	Name	hérité de: IdentifiedObject

6.3.20 Classe racine UserAttribute

Classe générique de paires nom-valeur avec numéro de séquence et unités facultatifs pour la valeur; peut être utilisée pour modéliser des parties de l'échange d'informations lorsque des types concrets ne sont pas connus à l'avance.

Le Tableau 35 montre tous les attributs de UserAttribute.

Tableau 35 – Attributs de Common::UserAttribute

nom	type	description
sequenceNumber	Integer	Numéro de séquence pour cet attribut dans une liste d'attributs.
name	String	Nom d'un attribut.
value	StringQuantity	Valeur d'un attribut, y compris les informations sur les unités.

Le Tableau 36 montre toutes les extrémités d'associations de UserAttribute avec les autres classes.

Tableau 36 – Extrémités d'associations de Common::UserAttribute avec les autres classes

[mult à partir de]	[mult vers] nom	type	description
[0..*]	[0..1] Transaction	Transaction	Transaction pour laquelle cet instantané a été enregistré.

6.4 Paquetage Assets

6.4.1 Généralités

Ce paquetage contient le noyau des classes d'informations qui soutiennent des applications de gestion de biens qui traitent des aspects physiques et de durée de vie des différentes ressources du réseau (par opposition aux modèles de ressources de système électrique définis dans le paquetage IEC61970::Wires).

La Figure 32 montre le diagramme de classe AssetsInheritance.

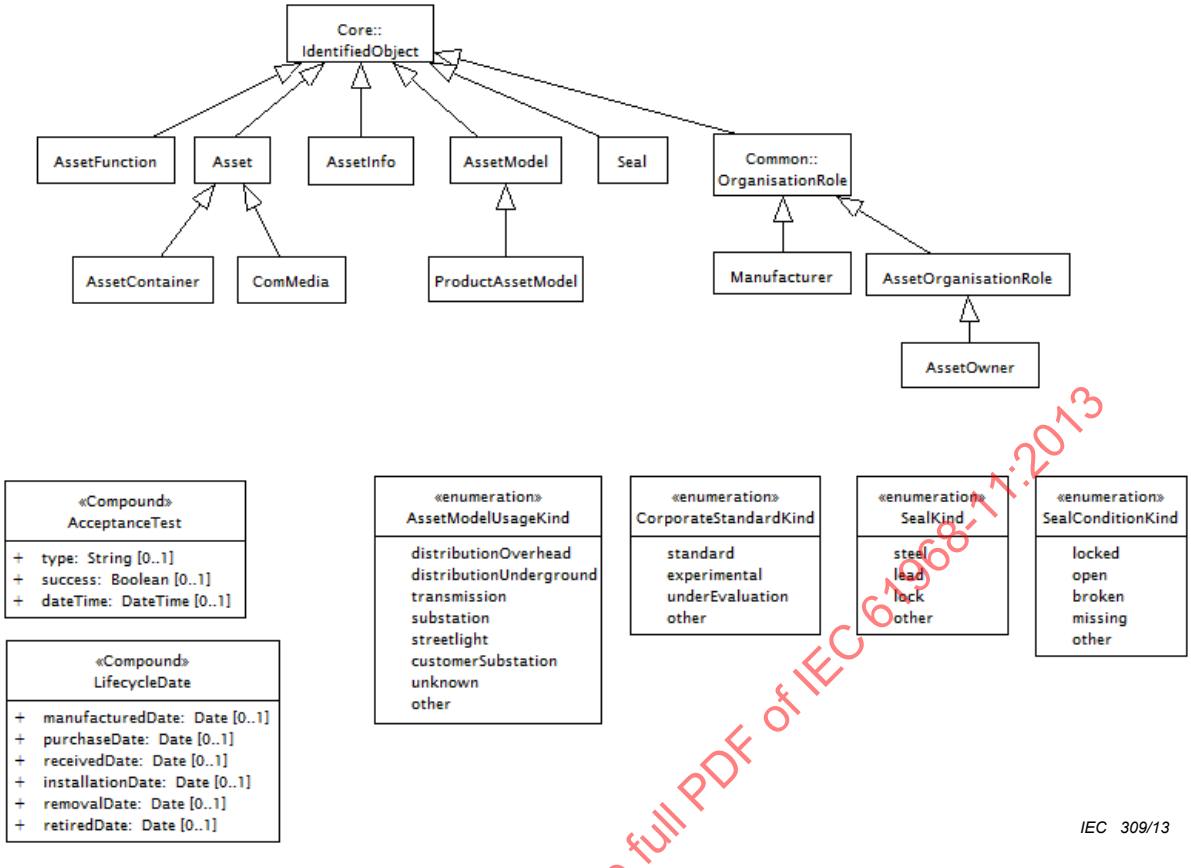


Figure 32 – Diagramme de classe Assets::AssetsInheritance

Ce diagramme montre la hiérarchie d'héritage pour les classes normatives issues de ce paquetage, ainsi que les énumérations et les types compound.

La Figure 33 montre le diagramme de classe AssetsOverview.

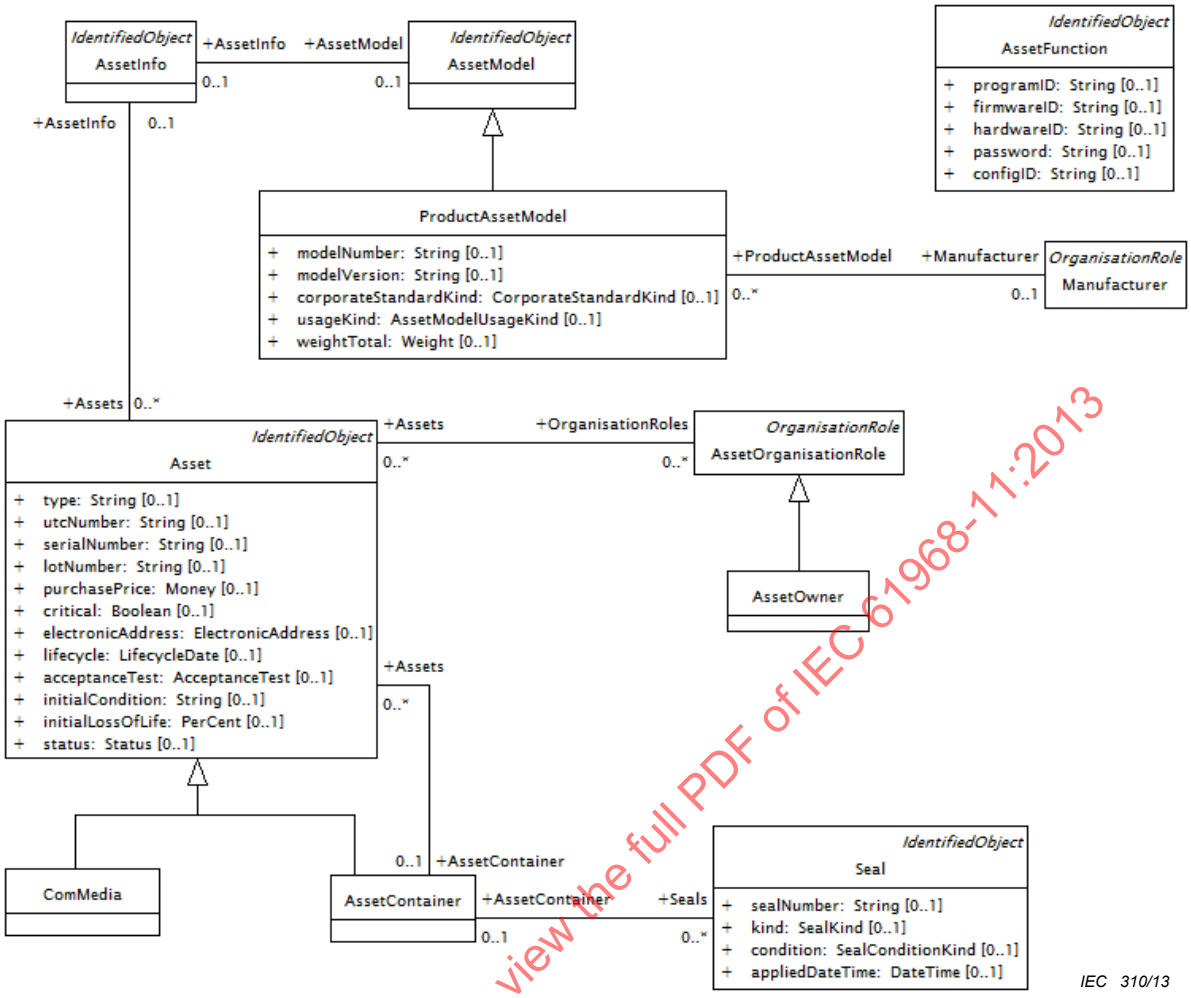


Figure 33 – Diagramme de classe Assets::AssetsOverview

Ce diagramme montre les classes normatives issues de ce paquetage.

6.4.2 Compound AcceptanceTest

Essai de réception pour les biens.

Le Tableau 37 montre tous les attributs de AcceptanceTest.

Tableau 37 – Attributs de Assets::AcceptanceTest

nom	type	description
type	String	Type d'essai ou de groupe d'essais qui a été réalisé à la date et heure 'dateTime'.
success	Boolean	True (c'est-à-dire vrai) si le bien a réussi à l'essai de réception et peut être mis en service ou y est déjà. Il est mis à false (c'est-à-dire faux) si le bien est retiré du service et nécessite d'être soumis de nouveau à essai avant d'être remis en service, éventuellement en un nouvel emplacement. Le bien pouvant subir plusieurs essais au cours de son cycle de vie, il est permis d'enregistrer la date de chaque essai de réception dans Asset.ActivityRecord.status.dateTime.
dateTime	DateTime	Date et heure auxquelles le bien a été soumis à essai la dernière fois en utilisant le 'type' d'essai et fournissant le statut courant dans l'attribut 'success'.

6.4.3 Compound LifecycleDate

Dates relatives aux événements de durée de vie d'un bien.

Le Tableau 38 montre tous les attributs de LifecycleDate.

Tableau 38 – Attributs de Assets::LifecycleDate

nom	type	description
manufacturedDate	Date	Date de fabrication du bien.
purchaseDate	Date	Date d'achat du bien. A noter que bien qu'un bien puisse avoir été acheté, il peut ne pas avoir été reçu dans l'inventaire au moment de l'achat.
receivedDate	Date	Date de réception et de première mise en place du bien dans l'inventaire.
installationDate	Date	(si applicable) Date à laquelle l'installation courante a été achevée, qui peut ne pas être la même que la date de mise en service. Le bien peut avoir été précédemment installé en d'autres emplacements. Ignoré si le bien (1) n'est pas actuellement installé (par exemple, stocké dans un dépôt) ou (2) n'est pas censé être installé (par exemple, véhicule, outil).
removalDate	Date	(si applicable) Date à laquelle le bien a été en dernier lieu retiré du service. Ignoré si (1) non destiné à être mis en service, ou (2) actuellement en service.
retiredDate	Date	(si applicable) Date à laquelle le bien est définitivement retiré du service et peut être programmé pour son élimination. Ignoré si le bien est (1) actuellement en service, ou (2) définitivement retiré du service.

6.4.4 Enumération AssetModelUsageKind

Usage pour un modèle de biens.

Le Tableau 39 montre tous les libellés de AssetModelUsageKind.

Tableau 39 – Libellés de Assets::AssetModelUsageKind

libellé	description
distributionOverhead	Le modèle de biens est destiné à être utilisé dans un réseau aérien de distribution.
distributionUnderground	Le modèle de biens est destiné à être utilisé dans un réseau de distribution souterrain.
transmission	Le modèle de biens est destiné à être utilisé dans un réseau de transmission.
substation	Le modèle de biens est destiné à être utilisé dans un poste.
streetlight	Le modèle de biens est destiné à être utilisé comme réverbère.
customerSubstation	Le modèle de biens est destiné à être utilisé dans un poste client.
unknown	L'usage du modèle de biens est inconnu.
other	Autre sorte d'usage de modèle de biens.

6.4.5 Enumération CorporateStandardKind

Sorte de norme d'entreprise.

Le Tableau 40 montre tous les libellés de CorporateStandardKind.

Tableau 40 – Libellés de Assets::CorporateStandardKind

libellé	description
standard	Le modèle de biens est utilisé comme norme d'entreprise.
experimental	Le modèle de biens est utilisé à titre expérimental.
underEvaluation	L'usage du modèle de biens est en cours d'évaluation.
other	Autre sorte de norme d'entreprise pour le modèle de biens.

6.4.6 Enumération SealConditionKind

Sorte de l'état du scellé.

Le Tableau 41 montre tous les libellés de SealConditionKind.

Tableau 41 – Libellés de Assets::SealConditionKind

libellé	description
locked	Le scellé est verrouillé.
open	Le scellé est ouvert.
broken	Le scellé est cassé.
missing	Le scellé est manquant.
Other (c'est-à-dire autre)	Autre sorte d'état du scellé.

6.4.7 Enumération SealKind

Sorte de scellé.

Le Tableau 42 montre tous les libellés de SealKind.

Tableau 42 – Libellés de Assets::SealKind

libellé	description
steel	Scellé d'acier.
lead	Scellé de plomb.
lock	Scellé de verrouillage.
other (c'est-à-dire autre)	Autre sorte de scellé.

6.4.8 Asset

Ressource tangible du service public, y compris l'équipement du système électrique, les divers dispositifs finaux, les armoires, les bâtiments, etc. Pour l'équipement du réseau électrique, le rôle du bien est défini par le truchement de PowerSystemResource et de ses sous-classes, définies principalement dans le modèle Wires (se référer à la CEI 61970-301 et au paquetage de modèle IEC61970::Wires). La description du bien met l'accent sur les caractéristiques physiques de l'équipement remplissant le rôle en question.

Le Tableau 43 montre tous les attributs de Asset.

Tableau 43 – Attributs de Assets::Asset

nom	type	description
type	String	Catégorisation spécifique aux entreprises de service public de Asset et de ses sous-types, en fonction de leurs normes d'entreprise, de leurs pratiques et de leurs systèmes TI existants (par exemple, pour la gestion des biens, de la maintenance, des travaux, des interruptions de service, des clients, etc.)
utcNumber	String	Numéro de Uniquely Tracked Commodity (UTC, c'est-à-dire marchandise localisée de manière unique).
serialNumber	String	Numéro de série de ce bien.
lotNumber	String	Numéro de lot de ce bien. Même pour le même modèle et le même numéro de version, de nombreux biens sont fabriqués par lots.
purchasePrice	Money	Prix d'achat du bien.
critical	Boolean	True (c'est-à-dire vrai) si le bien est considéré critique pour une certaine raison (par exemple, un poteau avec des fixations critiques).
electronicAddress	ElectronicAddress	Adresse électronique.
lifecycle	LifecycleDate	dates de durée de vie de ce bien.
acceptanceTest	AcceptanceTest	Informations relatives à l'essai de réception.
initialCondition	String	État du bien dans le stock ou au moment de l'installation. Les exemples sont notamment: neuf, rénové, révision complète requise, autre. Se référer aux données d'inspection pour avoir les informations sur l'état le plus actuel du bien.
initialLossOfLife	PerCent	Chaque fois qu'un bien est reconditionné, pourcentage de l'espérance de vie pour le bien lorsqu'il était neuf; zéro pour les dispositifs neufs.
status	Status	Statut de ce bien.
aliasName	String	hérité de: IdentifiedObject
mRID	String	hérité de: IdentifiedObject
name	String	hérité de: IdentifiedObject

Le Tableau 44 montre toutes les extrémités d'associations de Asset avec les autres classes.

Tableau 44 – Extrémités d'associations de Assets::Asset avec les autres classes

[mult à partir de]	[mult vers] nom	type	description
[0..*]	[0..*] PowerSystemResources	PowerSystemResource	Toutes les ressources du système électrique utilisées pour modéliser électriquement ce bien. Par exemple, le bien transformateur est modélisé électriquement avec un transformateur et ses enroulements et changeur de prise.
[0..*]	[0..*] ActivityRecords	ActivityRecord	Tous les enregistrements d'activité créés pour ce bien.
[0..1]	[0..*] ConfigurationEvents	ConfigurationEvent	Tous les événements de configuration créés pour ce bien.
[0..*]	[0..1] Location	Location	emplacement de ce bien.
[0..*]	[0..*] OrganisationRoles	AssetOrganisationRole	Tous les rôles qu'une organisation joue pour ce bien.
[0..*]	[0..1] AssetContainer	AssetContainer	Conteneur de ce bien.
[0..*]	[0..1] AssetInfo	AssetInfo	Données applicables à ce bien.

[mult à partir de]	[mult vers] nom	type	description
[0..1]	[0..*] DiagramObjects	DiagramObject	hérité de: IdentifiedObject
[1..1]	[0..*] Names	Name	hérité de: IdentifiedObject

6.4.9 AssetContainer

Bien qui est le regroupement d'autres biens tels que conducteurs, transformateurs, dispositifs de commutation, terrain, clôtures, bâtiments, équipements, véhicules, etc.

Le Tableau 45 montre tous les attributs de AssetContainer.

Tableau 45 – Attributs de Assets::AssetContainer

nom	type	description
type	String	hérité de: Asset
utcNumber	String	hérité de: Asset
serialNumber	String	hérité de: Asset
lotNumber	String	hérité de: Asset
purchasePrice	Money	hérité de: Asset
critical	Boolean	hérité de: Asset
electronicAddress	ElectronicAddress	hérité de: Asset
Lifecycle	LifecycleDate	hérité de: Asset
acceptanceTest	AcceptanceTest	hérité de: Asset
initialCondition	String	hérité de: Asset
initialLossOfLife	PerCent	hérité de: Asset
status	Status	hérité de: Asset
aliasName	String	hérité de: IdentifiedObject
mRID	String	hérité de: IdentifiedObject
name	String	hérité de: IdentifiedObject

Le Tableau 46 montre toutes les extrémités d'associations de AssetContainer avec les autres classes.

Tableau 46 – Extrémités d'associations de Assets::AssetContainer avec les autres classes

[mult à partir de]	[mult vers] nom	type	Description
[0..1]	[0..*] Seals	Seal	Tous les joints d'étanchéité appliqués à ce conteneur de biens.
[0..1]	[0..*] Assets	Asset	Tous les biens dans ce bien conteneur
[0..*]	[0..*] PowerSystemResources	PowerSystemResource	hérité de: Asset
[0..*]	[0..*] ActivityRecords	ActivityRecord	hérité de: Asset
[0..1]	[0..*] ConfigurationEvents	ConfigurationEvent	hérité de: Asset
[0..*]	[0..1] Location	Location	hérité de: Asset
[0..*]	[0..*] OrganisationRoles	AssetOrganisationRole	hérité de: Asset
[0..*]	[0..1] AssetContainer	AssetContainer	hérité de: Asset

[mult à partir de]	[mult vers] nom	type	Description
[0..*]	[0..1] AssetInfo	AssetInfo	hérité de: Asset
[0..1]	[0..*] DiagramObjects	DiagramObject	hérité de: IdentifiedObject
[1..1]	[0..*] Names	Name	hérité de: IdentifiedObject

6.4.10 AssetFunction

Fonction accomplie par un bien.

Le Tableau 47 montre tous les attributs de AssetFunction.

Tableau 47 – Attributs de Assets::AssetFunction

nom	type	Description
programID	String	Nom du programme.
firmwareID	String	Version du firmware.
hardwareID	String	Version de l'équipement matériel.
password	String	Mot de passe nécessaire pour accéder à cette fonction.
configID	String	Configuration spécifiée pour cette fonction.
aliasName	String	hérité de: IdentifiedObject
mRID	String	hérité de: IdentifiedObject
name	String	hérité de: IdentifiedObject

Le Tableau 48 montre toutes les extrémités d'associations de AssetFunction avec les autres classes.

Tableau 48 – Extrémités d'associations de Assets::AssetFunction avec les autres classes

[mult à partir de]	[mult vers] nom	type	Description
[0..1]	[0..*] DiagramObjects	DiagramObject	hérité de: IdentifiedObject
[1..1]	[0..*] Names	Name	hérité de: IdentifiedObject

6.4.11 AssetInfo

Jeu d'attributs d'un bien, représentant les informations types des feuilles de données d'un dispositif physique qui peut être instancié et partagé dans différents contextes d'échange de données:

– comme attributs d'une instance de bien (installée ou en stock)
– comme attributs d'un modèle de biens (produit d'un fabricant)
– comme attributs d'un bien type (type générique d'un bien tel qu'utilisé pour les/la conceptions/planification des extensions).

Le Tableau 49 montre tous les attributs de AssetInfo.

Tableau 49 – Attributs de Assets::AssetInfo

nom	type	Description
aliasName	String	hérité de: IdentifiedObject
mRID	String	hérité de: IdentifiedObject
name	String	hérité de: IdentifiedObject

Le Tableau 50 montre toutes les extrémités d'associations de AssetInfo avec les autres classes.

Tableau 50 – Extrémités d'associations de Assets::AssetInfo avec les autres classes

[mult à partir de]	[mult vers] nom	type	Description
[0..1]	[0..*] PowerSystemResource	PowerSystemResource	Toutes les ressources du système électrique avec ces informations de feuille de données.
[0..1]	[0..*] Assets	Asset	Tous les biens décrits par ces données.
[0..1]	[0..1] AssetModel	AssetModel	Modèle de biens décrit par ces données.
[0..1]	[0..*] DiagramObjects	DiagramObject	hérité de: IdentifiedObject
[1..1]	[0..*] Names	Name	hérité de: IdentifiedObject

6.4.12 AssetModel

Modèle d'un bien, soit un produit d'un fabricant particulier ou un modèle de biens ou élément de matériau générique. Les caractéristiques de feuille de données sont disponibles dans la sous-classe AssetInfo associée et peuvent être partagées avec les instances de biens ou de ressources du système électrique.

Le Tableau 51 montre tous les attributs de AssetModel.

Tableau 51 – Attributs de Assets::AssetModel

nom	type	Description
aliasName	String	hérité de: IdentifiedObject
mRID	String	hérité de: IdentifiedObject
name	String	hérité de: IdentifiedObject

Le Tableau 52 montre toutes les extrémités d'associations de AssetModel avec les autres classes.

Tableau 52 – Extrémités d'associations de Assets::AssetModel avec les autres classes

[mult à partir de]	[mult vers] nom	type	description
[0..1]	[0..1] AssetInfo	AssetInfo	Données applicables à ce modèle de biens.
[0..1]	[0..*] DiagramObjects	DiagramObject	hérité de: IdentifiedObject
[1..1]	[0..*] Names	Name	hérité de: IdentifiedObject

6.4.13 AssetOrganisationRole

Rôle qu'une organisation joue par rapport au bien.

Le Tableau 53 montre tous les attributs de AssetOrganisationRole.

Tableau 53 – Attributs de Assets::AssetOrganisationRole

nom	type	description
aliasName	String	hérité de: IdentifiedObject
mRID	String	hérité de: IdentifiedObject
name	String	hérité de: IdentifiedObject

Le Tableau 54 montre toutes les extrémités d'associations de AssetOrganisationRole avec les autres classes.

Tableau 54 – Extrémités d'associations de Assets::AssetOrganisationRole avec les autres classes

[mult à partir de]	[mult vers] nom	type	description
[0..*]	[0..*] Assets	Asset	Tous les biens applicables à ce rôle d'organisation.
[0..1]	[0..*] ConfigurationEvents	ConfigurationEvent	hérité de: OrganisationRole
[0..*]	[0..1] Organisation	Organisation	hérité de: OrganisationRole
[0..1]	[0..*] DiagramObjects	DiagramObject	hérité de: IdentifiedObject
[1..1]	[0..*] Names	Name	hérité de: IdentifiedObject

6.4.14 AssetOwner

Propriétaire du bien.

Le Tableau 55 montre tous les attributs de AssetOwner.

Tableau 55 – Attributs de Assets::AssetOwner

nom	type	description
aliasName	String	hérité de: IdentifiedObject
mRID	String	hérité de: IdentifiedObject
name	String	hérité de: IdentifiedObject

Le Tableau 56 montre toutes les extrémités d'associations de AssetOwner avec les autres classes.

Tableau 56 – Extrémités d'associations de Assets::AssetOwner avec les autres classes

[mult à partir de]	[mult vers] nom	type	description
[0..*]	[0..*] Assets	Asset	hérité de: AssetOrganisationRole
[0..1]	[0..*] ConfigurationEvents	ConfigurationEvent	hérité de: OrganisationRole
[0..*]	[0..1] Organisation	Organisation	hérité de: OrganisationRole
[0..1]	[0..*] DiagramObjects	DiagramObject	hérité de: IdentifiedObject
[1..1]	[0..*] Names	Name	hérité de: IdentifiedObject

6.4.15 ComMedia

Supports de communication tels que câble de fibre optique, ligne électrique, téléphone, etc.

Le Tableau 57 montre tous les attributs de ComMediaAsset.

Tableau 57 – Attributs de Assets::ComMediaAsset

nom	type	description
type	String	hérité de: Asset
utcNumber	String	hérité de: Asset
serialNumber	String	hérité de: Asset
lotNumber	String	hérité de: Asset
purchasePrice	Money	hérité de: Asset
critical	Boolean	hérité de: Asset
electronicAddress	ElectronicAddress	hérité de: Asset
lifecycle	LifecycleDate	hérité de: Asset
acceptanceTest	AcceptanceTest	hérité de: Asset
initialCondition	String	hérité de: Asset
initialLossOfLife	PerCent	hérité de: Asset
status	Status	hérité de: Asset
aliasName	String	hérité de: IdentifiedObject
mRID	String	hérité de: IdentifiedObject
name	String	hérité de: IdentifiedObject

Le Tableau 58 montre toutes les extrémités d'associations de ComMediaAsset avec les autres classes.

Tableau 58 – Extrémités d'associations de Assets::ComMediaAsset avec les autres classes

[mult à partir de]	[mult vers] nom	type	description
[0..*]	[0..*] PowerSystemResources	PowerSystemResource	hérité de: Asset
[0..*]	[0..*] ActivityRecords	ActivityRecord	hérité de: Asset
[0..1]	[0..*] ConfigurationEvents	ConfigurationEvent	hérité de: Asset
[0..*]	[0..1] Location	Location	hérité de: Asset
[0..*]	[0..*] OrganisationRoles	AssetOrganisationRole	hérité de: Asset

[mult à partir de]	[mult vers] nom	type	description
[0..*]	[0..1] AssetContainer	AssetContainer	hérité de: Asset
[0..*]	[0..1] AssetInfo	AssetInfo	hérité de: Asset
[0..1]	[0..*] DiagramObjects	DiagramObject	hérité de: IdentifiedObject
[1..1]	[0..*] Names	Name	hérité de: IdentifiedObject

6.4.16 Manufacturer

Organisation qui fabrique les produits du bien.

Le Tableau 59 montre tous les attributs de Manufacturer.

Tableau 59 – Attributs de Assets::Manufacturer

nom	type	description
aliasName	String	hérité de: IdentifiedObject
mRID	String	hérité de: IdentifiedObject
name	String	hérité de: IdentifiedObject

Le Tableau 60 montre toutes les extrémités d'associations de Manufacturer avec les autres classes.

Tableau 60 – Extrémités d'associations de Assets::Manufacturer avec les autres classes

[mult à partir de]	[mult vers] nom	type	description
[0..1]	[0..*] ProductAssetModel	ProductAssetModel	Tous les modèles de biens de ce fabricant.
[0..1]	[0..*] ConfigurationEvents	ConfigurationEvent	hérité de: OrganisationRole
[0..*]	[0..1] Organisation	Organisation	hérité de: OrganisationRole
[0..1]	[0..*] DiagramObjects	DiagramObject	hérité de: IdentifiedObject
[1..1]	[0..*] Names	Name	hérité de: IdentifiedObject

6.4.17 ProductAssetModel

Modèle de biens d'un fabricant spécifique.

Le Tableau 61 montre tous les attributs de ProductAssetModel.

Tableau 61 – Attributs de Assets::ProductAssetModel

nom	type	description
modelNumber	String	Numéro de modèle du fabricant.
modelVersion	String	Numéro de version pour le modèle de produit, qui indique le millésime du produit.
corporateStandardKind	CorporateStandardKind	Sorte de norme d'entreprise pour ce modèle de biens.
usageKind	AssetModelUsageKind	Usage prévu pour ce modèle de biens.
weightTotal	Weight	Poids total manufacturé du bien.

nom	type	description
aliasName	String	hérité de: IdentifiedObject
mRID	String	hérité de: IdentifiedObject
name	String	hérité de: IdentifiedObject

Le Tableau 62 montre toutes les extrémités d'associations de ProductAssetModel avec les autres classes.

Tableau 62 – Extrémités d'associations de Assets::ProductAssetModel avec les autres classes

[mult à partir de]	[mult vers] nom	type	description
[0..*]	[0..1] Manufacturer	Manufacturer	Fabricant de ce modèle de biens.
[0..1]	[0..1] AssetInfo	AssetInfo	hérité de: AssetModel
[0..1]	[0..*] DiagramObjects	DiagramObject	hérité de: IdentifiedObject
[1..1]	[0..*] Names	Name	hérité de: IdentifiedObject

6.4.18 Seal

Contrôle physiquement l'accès aux AssetContainers.

Le Tableau 63 montre tous les attributs de Seal.

Tableau 63 – Attributs de Assets::Seal

nom	type	description
sealNumber	String	(mot réservé) Numéro du scellé.
kind	SealKind	Sorte du scellé.
condition	SealConditionKind	État du scellé.
appliedDateTime	DateTime	Date et heures auxquelles ce scellé a été appliqué.
aliasName	String	hérité de: IdentifiedObject
mRID	String	hérité de: IdentifiedObject
name	String	hérité de: IdentifiedObject

Le Tableau 64 montre toutes les extrémités d'associations de Seal avec les autres classes.

Tableau 64 – Extrémités d'association de Assets::Seal avec les autres classes

[mult à partir de]	[mult vers] nom	type	description
[0..*]	[0..1] AssetContainer	AssetContainer	Conteneur de biens auxquels ce scellé est appliqué.
[0..1]	[0..*] DiagramObjects	DiagramObject	hérité de: IdentifiedObject
[1..1]	[0..*] Names	Name	hérité de: IdentifiedObject

6.5 Paquetage AssetInfo

6.5.1 Généralités

Ce paquetage est une extension du paquetage Assets et contient les classes noyau d'informations qui soutiennent la gestion des biens et les différentes applications de planification de tâches et de réseau avec des sous-classes AssetInfo spécialisées. Elles détiennent des attributs pouvant être référencés non seulement par des Asset ou AssetModel mais aussi par des ConductingEquipment.

La Figure 34 montre le diagramme de classe AssetInfoInheritance.

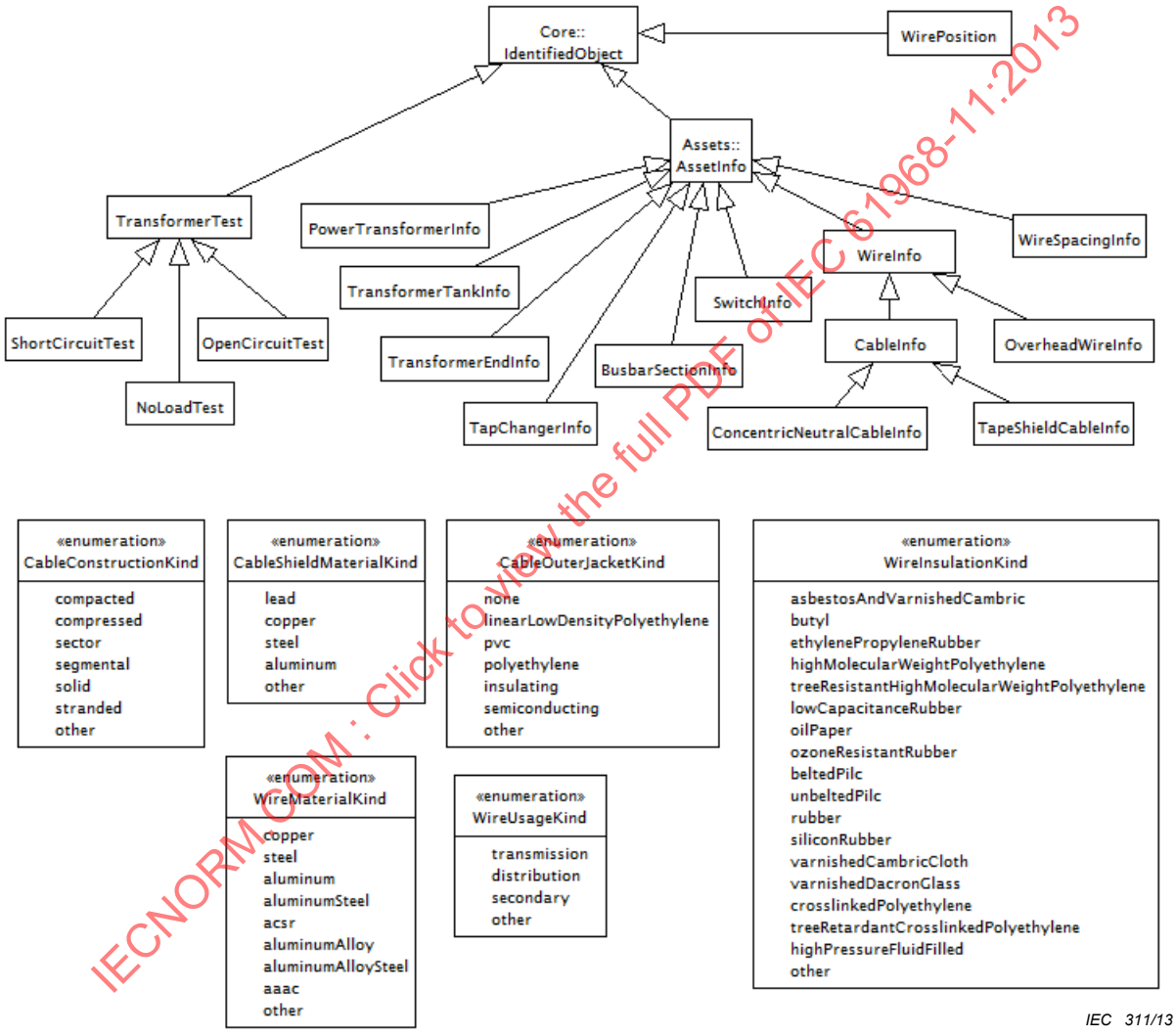


Figure 34 – Diagramme de classe AssetInfo::AssetInfoInheritance

Ce diagramme montre la hiérarchie d'héritage pour les classes normatives issues de ce paquetage, ainsi que les énumérations et les types compound.

La Figure 35 montre le diagramme de classe AssetInfoOverview.

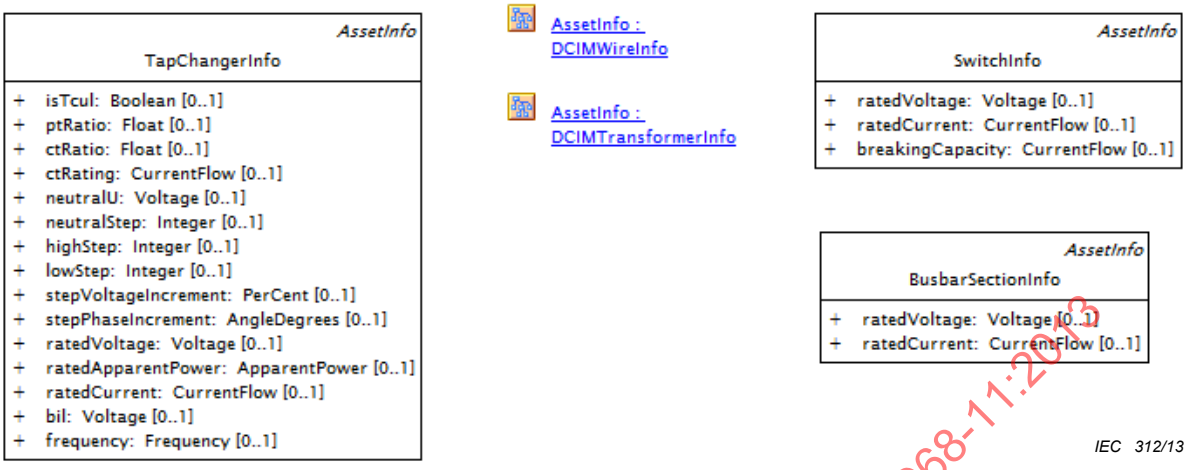


Figure 35 – Diagramme de classe AssetInfo::AssetInfoOverview

Ce diagramme montre les classes normatives issues de ce paquetage.

La Figure 36 montre le diagramme de classe DCIMWireInfo.

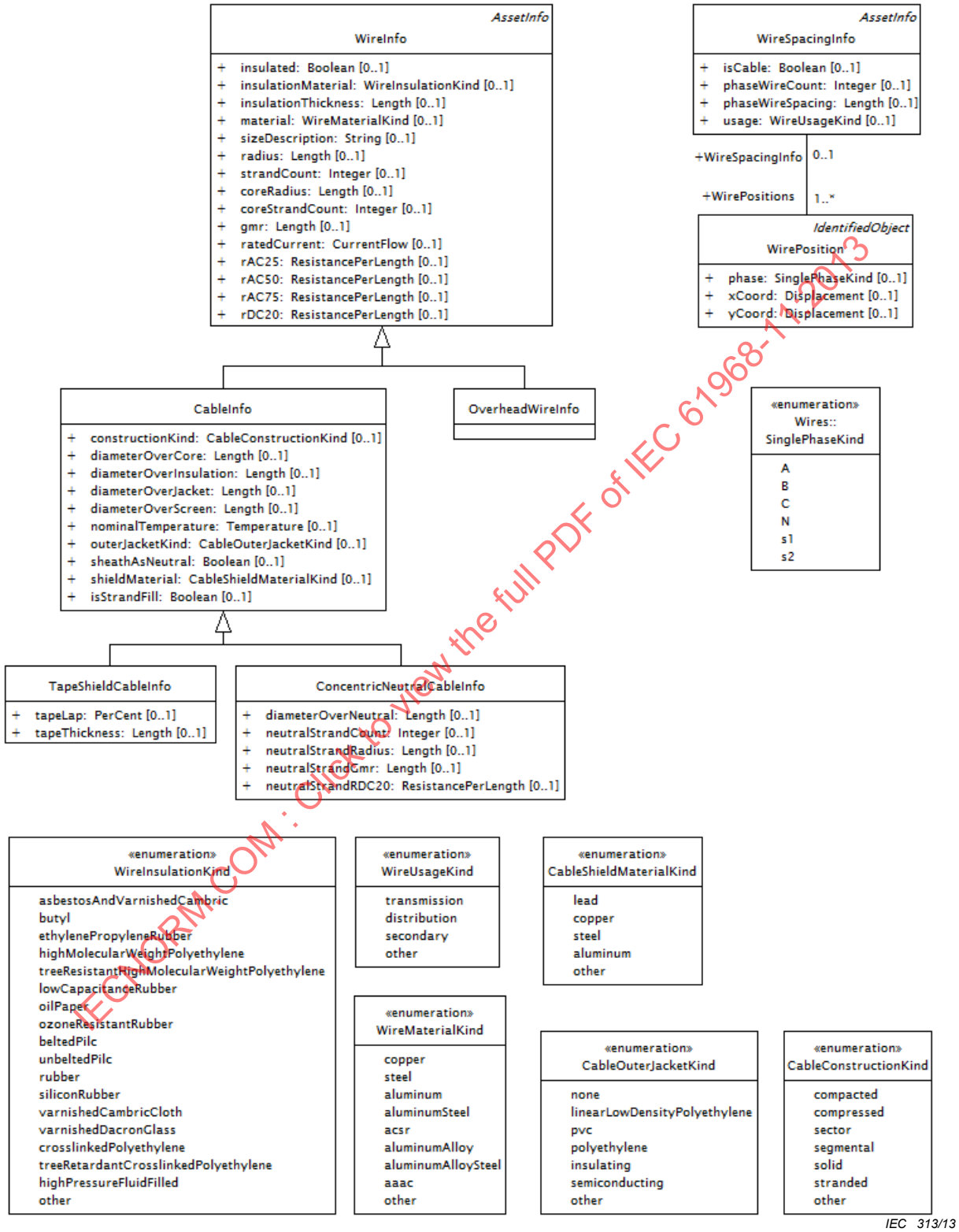


Figure 36 – Diagramme de classe AssetInfo::DCIMWireInfo

Ce diagramme montre les classes utilisées pour modéliser les aspects physiques des lignes dans le DCIM.

La Figure 37 montre le diagramme de classe DCIMTransformerInfo.

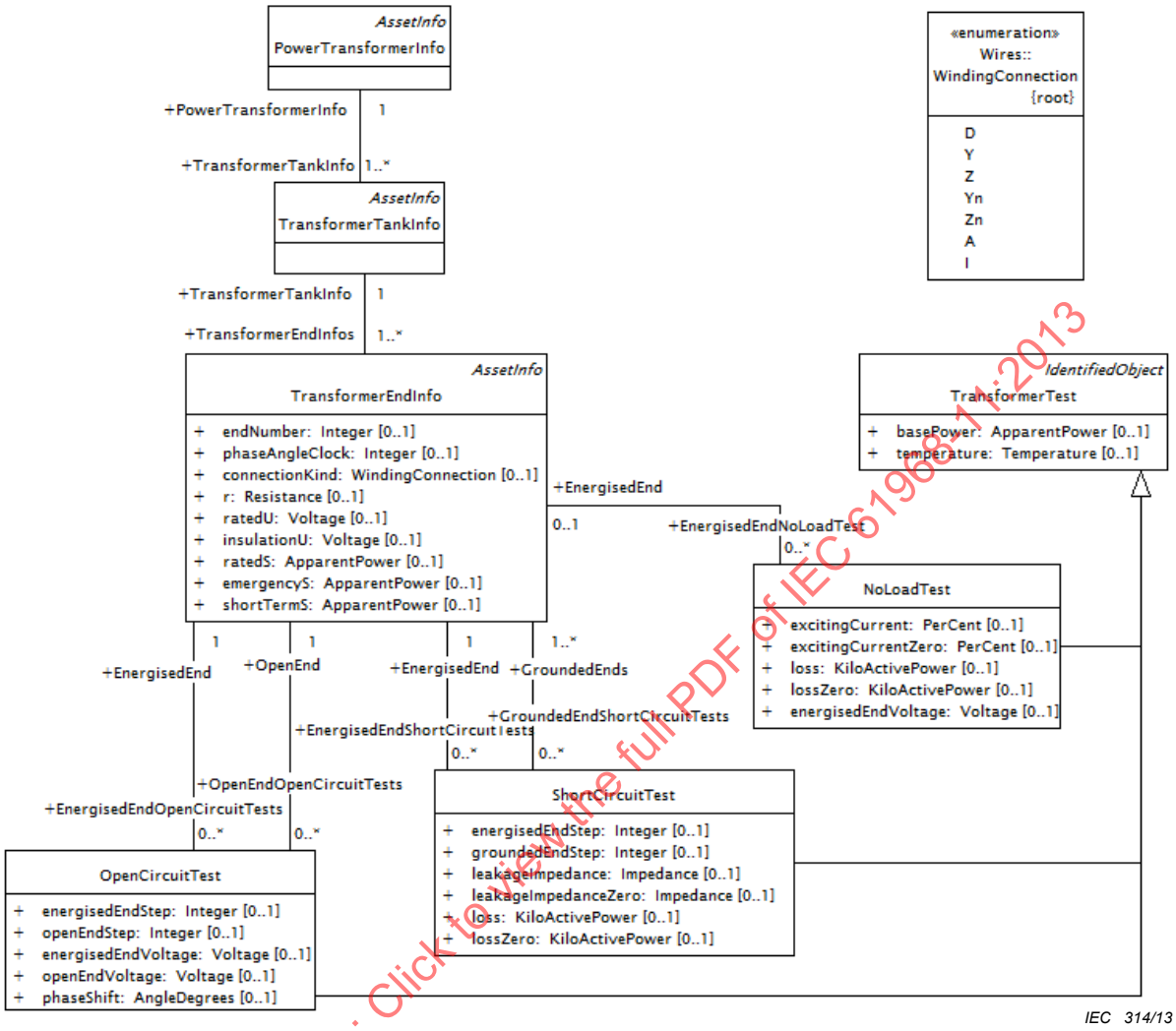


Figure 37 – Diagramme de classe AssetInfo::DCIMTransformerInfo

Ce diagramme montre une partie des classes utilisées pour modéliser les transformateurs dans le DCIM.

6.5.2 BusbarSectionInfo

Données relatives à la section de jeu de barres.

Le Tableau 65 montre tous les attributs de BusbarSectionInfo.

Tableau 65 – Attributs de AssetInfo::BusbarSectionInfo

nom	type	description
ratedVoltage	Voltage	Tension nominale.
ratedCurrent	CurrentFlow	Courant nominal.
aliasName	String	hérité de: IdentifiedObject
mRID	String	hérité de: IdentifiedObject
name	String	hérité de: IdentifiedObject

Le Tableau 66 montre toutes les extrémités d'associations de BusbarSectionInfo avec les autres classes.

Tableau 66 – Extrémités d'associations de AssetInfo::BusbarSectionInfo avec les autres classes

[mult à partir de]	[mult vers] nom	type	description
[0..1]	[0..*] PowerSystemResource	PowerSystemResource	hérité de: AssetInfo
[0..1]	[0..*] Assets	Asset	hérité de: AssetInfo
[0..1]	[0..1] AssetModel	AssetModel	hérité de: AssetInfo
[0..1]	[0..*] DiagramObjects	DiagramObject	hérité de: IdentifiedObject
[1..1]	[0..*] Names	Name	hérité de: IdentifiedObject

6.5.3 Enumération CableConstructionKind

Sorte de construction de câble.

Le Tableau 67 montre tous les libellés de CableConstructionKind.

Tableau 67 – Libellés de AssetInfo::CableConstructionKind

libellé	description
compacted	Câble compact.
compressed	Câble comprimé.
sector	Câble de secteur.
segmental	Câble segmentaire.
solid	Câble plein.
stranded	Câble toronné.
other	Autre sorte de construction de câble.

6.5.4 CableInfo

Données relatives aux câbles.

Le Tableau 68 montre tous les attributs de CableInfo.

Tableau 68 – Attributs de AssetInfo::CableInfo

nom	type	description
constructionKind	CableConstructionKind	Sorte de construction de ce câble.
diameterOverCore	Length	Diamètre sur l'âme, y compris l'écran semi-conducteur; il convient qu'il soit le diamètre intérieur de la couche isolante.
diameterOverInsulation	Length	Diamètre sur la couche isolante, écran extérieur exclu.
diameterOverJacket	Length	Diamètre sur la couche de gainage la plus extérieure.
diameterOverScreen	Length	Diamètre sur l'écran extérieur; il convient qu'il soit le diamètre intérieur du blindage.
nominalTemperature	Temperature	Valeur maximale nominale de la température de fonctionnement théorique.
outerJacketKind	CableOuterJacketKind	Sorte de chemise extérieure de ce câble.

nom	type	description
sheathAsNeutral	Boolean	True (c'est-à-dire vrai) si la gaine / le blindage sert de neutre (c'est-à-dire, est à la masse).
shieldMaterial	CableShieldMaterialKind	Matériau du blindage.
isStrandFill	Boolean	True (c'est-à-dire vrai) si les brins de fils sont extrudés de manière à remplir les vides dans le câble.
insulated	Boolean	hérité de: WireInfo
insulationMaterial	WireInsulationKind	hérité de: WireInfo
insulationThickness	Length	hérité de: WireInfo
material	WireMaterialKind	hérité de: WireInfo
sizeDescription	String	hérité de: WireInfo
radius	Length	hérité de: WireInfo
strandCount	Integer	hérité de: WireInfo
coreRadius	Length	hérité de: WireInfo
coreStrandCount	Integer	hérité de: WireInfo
gmr	Length	hérité de: WireInfo
ratedCurrent	CurrentFlow	hérité de: WireInfo
rAC25	ResistancePerLength	hérité de: WireInfo
rAC50	ResistancePerLength	hérité de: WireInfo
rAC75	ResistancePerLength	hérité de: WireInfo
rDC20	ResistancePerLength	hérité de: WireInfo
aliasName	String	hérité de: IdentifiedObject
mRID	String	hérité de: IdentifiedObject
name	String	hérité de: IdentifiedObject

Le Tableau 69 montre toutes les extrémités d'associations de CableInfo avec les autres classes.

Tableau 69 – Extrémités d'associations de AssetInfo::CableInfo avec les autres classes

[mult à partir de]	[mult vers] nom	type	description
[0..*]	[0..*] Impedances	PerLengthImpedance	hérité de: WireInfo
[0..1]	[0..*] PowerSystemResource	PowerSystemResource	hérité de: AssetInfo
[0..1]	[0..*] Assets	Asset	hérité de: AssetInfo
[0..1]	[0..1] AssetModel	AssetModel	hérité de: AssetInfo
[0..1]	[0..*] DiagramObjects	DiagramObject	hérité de: IdentifiedObject
[1..1]	[0..*] Names	Name	hérité de: IdentifiedObject

6.5.5 Enumération CableOuterJacketKind

Sorte de chemise extérieure de câble.

Le Tableau 70 montre tous les libellés de CableOuterJacketKind.

Tableau 70 – Libellés de AssetInfo::CableOuterJacketKind

libellé	description
none	Le câble n'a pas de chemise extérieure.
linearLowDensityPolyethylene	Chemise extérieure de câble en polyéthylène basse densité linéaire.
pvc	Chemise extérieure de câble en PVC.
polyethylene	Chemise extérieure de câble en polyéthylène.
insulating	Chemise extérieure de câble isolante.
semiconducting	Chemise extérieure de câble semi-conductrice.
other	Autre sorte de chemise extérieure de câble.

6.5.6 Enumération CableShieldMaterialKind

Sorte de matériau de blindage de câble.

Le Tableau 71 montre tous les libellés de CableShieldMaterialKind.

Tableau 71 – Libellés de AssetInfo::CableShieldMaterialKind

libellé	description
lead	Blindage de câble en plomb.
copper	Blindage de câble en cuivre.
steel	Blindage de câble en acier.
aluminum	Blindage de câble en aluminium
other	Autre sorte de matériau de blindage de câble

6.5.7 ConcentricNeutralCableInfo

Données relatives au câble à neutre concentrique.

Le Tableau 72 montre tous les attributs de ConcentricNeutralCableInfo.

Tableau 72 – Attributs de AssetInfo::ConcentricNeutralCableInfo

nom	type	description
diameterOverNeutral	Length	Diamètre sur les brins à neutre concentrique.
neutralStrandCount	Integer	Nombre de brins à neutre concentrique.
neutralStrandRadius	Length	Rayon extérieur du brin neutre.
neutralStrandGmr	Length	Rayon moyen géométrique du brin neutre.
neutralStrandRDC20	ResistancePerLength	Résistance CC linéique du brin neutre à 20 °C.
constructionKind	CableConstructionKind	hérité de: CableInfo
diameterOverCore	Length	hérité de: CableInfo
diameterOverInsulation	Length	hérité de: CableInfo
diameterOverJacket	Length	hérité de: CableInfo
diameterOverScreen	Length	hérité de: CableInfo
nominalTemperature	Temperature	hérité de: CableInfo

nom	type	description
outerJacketKind	CableOuterJacketKind	hérité de: CableInfo
sheathAsNeutral	Boolean	hérité de: CableInfo
shieldMaterial	CableShieldMaterialKind	hérité de: CableInfo
isStrandFill	Boolean	hérité de: CableInfo
insulated	Boolean	hérité de: WireInfo
insulationMaterial	WireInsulationKind	hérité de: WireInfo
insulationThickness	Length	hérité de: WireInfo
material	WireMaterialKind	hérité de: WireInfo
sizeDescription	String	hérité de: WireInfo
radius	Length	hérité de: WireInfo
strandCount	Integer	hérité de: WireInfo
coreRadius	Length	hérité de: WireInfo
coreStrandCount	Integer	hérité de: WireInfo
gmr	Length	hérité de: WireInfo
ratedCurrent	CurrentFlow	hérité de: WireInfo
rAC25	ResistancePerLength	hérité de: WireInfo
rAC50	ResistancePerLength	hérité de: WireInfo
rAC75	ResistancePerLength	hérité de: WireInfo
rDC20	ResistancePerLength	hérité de: WireInfo
aliasName	String	hérité de: IdentifiedObject
mRID	String	hérité de: IdentifiedObject
name	String	hérité de: IdentifiedObject

Le Tableau 73 montre toutes les extrémités d'associations de ConcentricNeutralCableInfo avec les autres classes.

Tableau 73 – Extrémités d'association de AssetInfo::ConcentricNeutralCableInfo avec les autres classes

[mult à partir de]	[mult vers] nom	type	description
[0..*]	[0..*] Impedances	PerLengthImpedance	hérité de: WireInfo
[0..1]	[0..*] PowerSystemResource	PowerSystemResource	hérité de: AssetInfo
[0..1]	[0..*] Assets	Asset	hérité de: AssetInfo
[0..1]	[0..1] AssetModel	AssetModel	hérité de: AssetInfo
[0..1]	[0..*] DiagramObjects	DiagramObject	hérité de: IdentifiedObject
[1..1]	[0..*] Names	Name	hérité de: IdentifiedObject

6.5.8 NoLoadTest

Les résultats de l'essai à vide déterminent les paramètres d'admittance de noyau. Ils comprennent les mesures du courant d'excitation et de la perte dans le fer par application de la tension à un enroulement. L'excitation peut être directe ou homopolaire. L'essai peut être répété à différentes tensions afin de mesurer la saturation.

Le Tableau 74 montre tous les attributs de NoLoadTest.

Tableau 74 – Attributs de AssetInfo::NoLoadTest

nom	type	description
excitingCurrent	PerCent	Courant d'excitation mesuré à partir d'un essai d'excitation de circuit ouvert monophasé ou direct.
excitingCurrentZero	PerCent	Courant d'excitation mesuré à partir d'un essai d'excitation de circuit ouvert homopolaire.
loss	KiloActivePower	Pertes mesurées à partir d'un essai d'excitation de circuit ouvert monophasé ou direct.
lossZero	KiloActivePower	Pertes mesurées à partir d'un essai d'excitation homopolaire.
energisedEndVoltage	Voltage	Tension appliquée à l'enroulement (extrémité) pendant l'essai.
basePower	ApparentPower	hérité de: TransformerTest
temperature	Temperature	hérité de: TransformerTest
aliasName	String	hérité de: IdentifiedObject
mRID	String	hérité de: IdentifiedObject
name	String	hérité de: IdentifiedObject

Le Tableau 75 montre toutes les extrémités d'associations de NoLoadTest avec les autres classes.

Tableau 75 – Extrémités d'associations de AssetInfo::NoLoadTest avec les autres classes

[mult à partir de]	[mult vers] nom	type	description
[0..*]	[0..1] EnergisedEnd	TransformerEndInfo	Extrémité de transformateur à laquelle le courant est appliqué pour cet essai à vide.
[0..1]	[0..*] DiagramObjects	DiagramObject	hérité de: IdentifiedObject
[1..1]	[0..*] Names	Name	hérité de: IdentifiedObject

6.5.9 OpenCircuitTest

Les résultats d'essai de circuit ouvert vérifient les rapports d'enroulements et les déphasages d'enroulement. Ils comprennent les mesures de la tension induite et du déphasage sur les enroulements en circuit ouvert, avec application de la tension à l'extrémité mise sous tension. Pour les enroulements triphasés, l'excitation peut être directe (la valeur par défaut) ou homopolaire.

Le Tableau 76 montre tous les attributs de OpenCircuitTest.

Tableau 76 – Attributs de AssetInfo::OpenCircuitTest

nom	type	description
energisedEndStep	Integer	Nombre de régleurs de l'enroulement pour l'extrémité mise sous tension de la paire d'essai.
openEndStep	Integer	Nombre de régleurs de l'enroulement pour l'extrémité ouverte de la paire d'essai.

nom	type	description
energisedEndVoltage	Voltage	Tension appliquée à l'enroulement (extrémité) pendant l'essai.
openEndVoltage	Voltage	Tension mesurée sur l'extrémité en circuit ouvert, l'extrémité mise sous tension étant à la tension nominale et toutes les autres en circuit ouvert.
phaseShift	AngleDegrees	Déphasage mesuré sur l'extrémité ouverte, l'extrémité mise sous tension étant à la tension nominale et toutes les autres en circuit ouvert.
basePower	ApparentPower	hérité de: TransformerTest
temperature	Temperature	hérité de: TransformerTest
aliasName	String	hérité de: IdentifiedObject
mRID	String	hérité de: IdentifiedObject
name	String	hérité de: IdentifiedObject

Le Tableau 77 montre toutes les extrémités d'associations de OpenCircuitTest avec les autres classes.

Tableau 77 – Extrémités d'associations de AssetInfo::OpenCircuitTest avec les autres classes

[mult à partir de]	[mult vers] nom	type	description
[0..*]	[1..1] OpenEnd	TransformerEndInfo	Extrémité de transformateur mesurée pour la tension et l'angle induits dans cet essai de circuit ouvert.
[0..*]	[1..1] EnergisedEnd	TransformerEndInfo	Extrémité de transformateur à laquelle le courant est appliqué dans cet essai de circuit ouvert.
[0..1]	[0..*] DiagramObjects	DiagramObject	hérité de: IdentifiedObject
[1..1]	[0..*] Names	Name	hérité de: IdentifiedObject

6.5.10 OverheadWireInfo

Données relatives au conducteur aérien.

Le Tableau 78 montre tous les attributs de OverheadWireInfo.

Tableau 78 – Attributs de AssetInfo::OverheadWireInfo

nom	type	description
insulated	Boolean	hérité de: WireInfo
insulationMaterial	WireInsulationKind	hérité de: WireInfo
insulationThickness	Length	hérité de: WireInfo
material	WireMaterialKind	hérité de: WireInfo
sizeDescription	String	hérité de: WireInfo
radius	Length	hérité de: WireInfo
strandCount	Integer	hérité de: WireInfo
coreRadius	Length	hérité de: WireInfo
coreStrandCount	Integer	hérité de: WireInfo
gmr	Length	hérité de: WireInfo

nom	type	description
ratedCurrent	CurrentFlow	hérité de: WireInfo
rAC25	ResistancePerLength	hérité de: WireInfo
rAC50	ResistancePerLength	hérité de: WireInfo
rAC75	ResistancePerLength	hérité de: WireInfo
rDC20	ResistancePerLength	hérité de: WireInfo
aliasName	String	hérité de: IdentifiedObject
mRID	String	hérité de: IdentifiedObject
name	String	hérité de: IdentifiedObject

Le Tableau 79 montre toutes les extrémités d'associations de OverheadWireInfo avec les autres classes.

Tableau 79 – Extrémités d'association de AssetInfo::OverheadWireInfo avec les autres classes

[mult à partir de]	[mult vers] nom	type	description
[0..*]	[0..*] Impedances	PerLengthImpedance	hérité de: WireInfo
[0..1]	[0..*] PowerSystemResource	PowerSystemResource	hérité de: AssetInfo
[0..1]	[0..*] Assets	Asset	hérité de: AssetInfo
[0..1]	[0..1] AssetModel	AssetModel	hérité de: AssetInfo
[0..1]	[0..*] DiagramObjects	DiagramObject	hérité de: IdentifiedObject
[1..1]	[0..*] Names	Name	hérité de: IdentifiedObject

6.5.11 PowerTransformerInfo

Jeu de données relatives au transformateur de puissance, issues d'une bibliothèque d'équipement.

Le Tableau 80 montre tous les attributs de PowerTransformerInfo.

Tableau 80 – Attributs de AssetInfo::PowerTransformerInfo

nom	type	description
aliasName	String	hérité de: IdentifiedObject
mRID	String	hérité de: IdentifiedObject
name	String	hérité de: IdentifiedObject

Le Tableau 81 montre toutes les extrémités d'associations de PowerTransformerInfo avec les autres classes.

Tableau 81 – Extrémités d'associations de AssetInfo::PowerTransformerInfo avec les autres classes

[mult à partir de]	[mult vers] nom	type	description
[1..1]	[1..*] TransformerTankInfo	TransformerTankInfo	Données pour toutes les cuves décrites par ces données relatives au transformateur de puissance.
[0..1]	[0..*] PowerSystemResource	PowerSystemResource	hérité de: AssetInfo
[0..1]	[0..*] Assets	Asset	hérité de: AssetInfo
[0..1]	[0..1] AssetModel	AssetModel	hérité de: AssetInfo
[0..1]	[0..*] DiagramObjects	DiagramObject	hérité de: IdentifiedObject
[1..1]	[0..*] Names	Name	hérité de: IdentifiedObject

6.5.12 ShortCircuitTest

Les résultats d'essai de court-circuit déterminent les paramètres d'impédance de maille. Ils comprennent les pertes par les charges et les impédances de fuite. Pour les enroulements triphasés, l'excitation peut être directe (la valeur par défaut) ou homopolaire. Il doit y avoir au moins un enroulement mis à la terre.

Le Tableau 82 montre tous les attributs de ShortCircuitTest.

Tableau 82 – Attributs de AssetInfo::ShortCircuitTest

nom	type	description
energisedEndStep	Integer	Numéro de la prise de réglage de l'enroulement pour l'extrémité mise sous tension de la paire d'essai.
groundedEndStep	Integer	Numéro de la prise de réglage de l'enroulement pour l'extrémité mise à la terre de la paire d'essai.
leakageImpedance	Impedance	Impédance de fuite mesurée à partir d'un essai de court-circuit monophasé ou direct.
leakageImpedanceZero	Impedance	Impédance de fuite mesurée à partir d'un essai de court-circuit homopolaire.
loss	KiloActivePower	Pertes par les charges à partir d'un essai de court-circuit monophasé ou direct.
lossZero	KiloActivePower	Pertes par les charges à partir d'un essai de court-circuit homopolaire.
basePower	ApparentPower	hérité de: TransformerTest
temperature	Temperature	hérité de: TransformerTest
aliasName	String	hérité de: IdentifiedObject
mRID	String	hérité de: IdentifiedObject
name	String	hérité de: IdentifiedObject

Le Tableau 83 montre toutes les extrémités d'associations de ShortCircuitTest avec les autres classes.

Tableau 83 – Extrémités d'associations de AssetInfo::ShortCircuitTest avec les autres classes

[mult à partir de]	[mult vers] nom	type	description
[0..*]	[1..*] GroundedEnds	TransformerEndInfo	Toutes les extrémités en court-circuit au cours de cet essai de court-circuit.
[0..*]	[1..1] EnergisedEnd	TransformerEndInfo	Extrémité de transformateur à laquelle la tension est appliquée dans cet essai de court-circuit. La tension d'essai est choisie pour induire le courant nominal dans l'extrémité mise sous tension.
[0..1]	[0..*] DiagramObjects	DiagramObject	hérité de: IdentifiedObject
[1..1]	[0..*] Names	Name	hérité de: IdentifiedObject

6.5.13 SwitchInfo

Données relatives au Switch (organe de coupure – commutateur).

Le Tableau 84 montre tous les attributs de SwitchInfo.

Tableau 84 – Attributs de AssetInfo::SwitchInfo

nom	type	description
ratedVoltage	Voltage	Tension nominale.
ratedCurrent	CurrentFlow	Courant nominal.
breakingCapacity	CurrentFlow	Courant de défaut maximal auquel un dispositif de coupure peut fonctionner en toute sécurité dans les conditions d'utilisation spécifiées.
aliasName	String	hérité de: IdentifiedObject
mRID	String	hérité de: IdentifiedObject
name	String	hérité de: IdentifiedObject

Le Tableau 85 montre toutes les extrémités d'associations de SwitchInfo avec les autres classes.

Tableau 85 – Extrémités d'associations de AssetInfo::SwitchInfo avec les autres classes

[mult à partir de]	[mult vers] nom	type	description
[0..1]	[0..*] PowerSystemResource	PowerSystemResource	hérité de: AssetInfo
[0..1]	[0..*] Assets	Asset	hérité de: AssetInfo
[0..1]	[0..1] AssetModel	AssetModel	hérité de: AssetInfo
[0..1]	[0..*] DiagramObjects	DiagramObject	hérité de: IdentifiedObject
[1..1]	[0..*] Names	Name	hérité de: IdentifiedObject

6.5.14 TapChangerInfo

Données relatives aux changeurs de prise.

Le Tableau 86 montre tous les attributs de TapChangerInfo.

Tableau 86 – Attributs de AssetInfo::TapChangerInfo

nom	type	description
isTcul	Boolean	Si ce changeur de prise dispose ou non des capacités de changeur de prise en charge.
ptRatio	Float	Rapport de transformation de transducteur intégré.
ctRatio	Float	Rapport des courants de transducteur intégré.
ctRating	CurrentFlow	Caractéristique nominale de primaire de transformateur de courant intégré.
neutralU	Voltage	Tension à laquelle l'enroulement fonctionne au réglage de la prise neutre.
neutralStep	Integer	Position de la prise de réglage neutre pour l'enroulement.
highStep	Integer	Plus haute position possible du régleur, avance par rapport au neutre.
lowStep	Integer	Plus basse position possible du régleur, retard par rapport au neutre.
stepVoltageIncrement	PerCent	Incrément de régleur, en pourcentage de la tension nominale, par position de régleur.
stepPhaseIncrement	AngleDegrees	Déphasage par position de régleur.
ratedVoltage	Voltage	Tension nominale.
ratedApparentPower	ApparentPower	Puissance apparente nominale.
ratedCurrent	CurrentFlow	Courant nominal.
bil	Voltage	Niveau d'isolement de base (BIL) exprimé en tension maximale de choc d'une onde nominale, généralement 1,2 X 50 µs. Il s'agit de la mesure de la capacité de l'isolation à résister à de fortes surtensions.
frequency	Frequency	Fréquence à laquelle les caractéristiques nominales s'appliquent.
aliasName	String	hérité de: IdentifiedObject
mRID	String	hérité de: IdentifiedObject
name	String	hérité de: IdentifiedObject

Le Tableau 87 montre toutes les extrémités d'associations de TapChangerInfo avec les autres classes.

Tableau 87 – Extrémités d'associations de AssetInfo::TapChangerInfo avec les autres classes

[mult à partir de]	[mult vers] nom	type	description
[0..1]	[0..*] PowerSystemResource	PowerSystemResource	hérité de: AssetInfo
[0..1]	[0..*] Assets	Asset	hérité de: AssetInfo
[0..1]	[0..1] AssetModel	AssetModel	hérité de: AssetInfo
[0..1]	[0..*] DiagramObjects	DiagramObject	hérité de: IdentifiedObject
[1..1]	[0..*] Names	Name	hérité de: IdentifiedObject

6.5.15 TapeShieldCableInfo

Données relatives au câble à blindage en ruban.

Le Tableau 88 montre tous les attributs de TapeShieldCableInfo.

Tableau 88 – Attributs de AssetInfo::TapeShieldCableInfo

nom	type	description
tapeLap	PerCent	Pourcentage de la largeur du blindage en ruban qui se chevauche dans chaque enveloppement, typiquement 10 % à 25 %.
tapeThickness	Length	Épaisseur du blindage en ruban, avant l'enrubannage.
constructionKind	CableConstructionKind	hérité de: CableInfo
diameterOverCore	Length	hérité de: CableInfo
diameterOverInsulation	Length	hérité de: CableInfo
diameterOverJacket	Length	hérité de: CableInfo
diameterOverScreen	Length	hérité de: CableInfo
nominalTemperature	Temperature	hérité de: CableInfo
outerJacketKind	CableOuterJacketKind	hérité de: CableInfo
sheathAsNeutral	Boolean	hérité de: CableInfo
shieldMaterial	CableShieldMaterialKind	hérité de: CableInfo
isStrandFill	Boolean	hérité de: CableInfo
insulated	Boolean	hérité de: WireInfo
insulationMaterial	WireInsulationKind	hérité de: WireInfo
insulationThickness	Length	hérité de: WireInfo
material	WireMaterialKind	hérité de: WireInfo
sizeDescription	String	hérité de: WireInfo
radius	Length	hérité de: WireInfo
strandCount	Integer	hérité de: WireInfo
coreRadius	Length	hérité de: WireInfo
coreStrandCount	Integer	hérité de: WireInfo
gmr	Length	hérité de: WireInfo
ratedCurrent	CurrentFlow	hérité de: WireInfo
rAC25	ResistancePerLength	hérité de: WireInfo
rAC50	ResistancePerLength	hérité de: WireInfo
rAC75	ResistancePerLength	hérité de: WireInfo
rDC20	ResistancePerLength	hérité de: WireInfo
aliasName	String	hérité de: IdentifiedObject
mRID	String	hérité de: IdentifiedObject
name	String	hérité de: IdentifiedObject

Le Tableau 89 montre toutes les extrémités d'associations de TapeShieldCableInfo avec les autres classes.

Tableau 89 – Extrémités d'associations de AssetInfo::TapeShieldCableInfo avec les autres classes

[mult à partir de]	[mult vers] nom	type	description
[0..*]	[0..*] Impedances	PerLengthImpedance	hérité de: WireInfo
[0..1]	[0..*] PowerSystemResource	PowerSystemResource	hérité de: AssetInfo
[0..1]	[0..*] Assets	Asset	hérité de: AssetInfo
[0..1]	[0..1] AssetModel	AssetModel	hérité de: AssetInfo
[0..1]	[0..*] DiagramObjects	DiagramObject	hérité de: IdentifiedObject
[1..1]	[0..*] Names	Name	hérité de: IdentifiedObject

6.5.16 TransformerEndInfo

Données relatives aux extrémités de transformateur.

Le Tableau 90 montre tous les attributs de TransformerEndInfo.

Tableau 90 – Attributs de AssetInfo::TransformerEndInfo

nom	type	description
endNumber	Integer	Numéro de cette extrémité de transformateur, correspondant à l'ordre de l'extrémité dans l'attribut PowerTransformer.vectorGroup. Il convient que l'enroulement de plus haute tension soit le 1.
phaseAngleClock	Integer	Angle de phase d'enroulement où 360 degrés sont représentés par des heures d'horloge et, donc, les valeurs valides sont {0, ..., 11}. Par exemple, pour exprimer le code d'enroulement 'Dyn11', régler les attributs comme suit: 'endNumber'=2, 'connectionKind' = Yn et 'phaseAngleClock' = 11.
connectionKind	WindingConnection	Sorte de connexion.
r	Resistance	Résistance CC.
ratedU	Voltage	Tension nominale: entre phases pour les enroulements triphasés et soit entre phases, soit entre phase et neutre pour les enroulements monophasés.
insulationU	Voltage	Caractéristique nominale de tension au niveau d'isolement de base.
ratedS	ApparentPower	Caractéristique nominale de puissance apparente normale.
emergencyS	ApparentPower	Puissance apparente pouvant être supportée par l'enroulement dans des conditions d'urgence (également appelée puissance d'urgence de longue durée).
shortTermS	ApparentPower	Puissance apparente pouvant être supportée par cet enroulement sur une courte période (en condition d'urgence).
aliasName	String	hérité de: IdentifiedObject
mRID	String	hérité de: IdentifiedObject
name	String	hérité de: IdentifiedObject

Le Tableau 91 montre toutes les extrémités d'associations de TransformerEndInfo avec les autres classes.

Tableau 91 – Extrémités d'associations de AssetInfo::TransformerEndInfo avec les autres classes

[mult à partir de]	[mult vers] nom	type	description
[0..1]	[0..1] CoreAdmittance	TransformerCoreAdmittance	Admittance de noyau calculée à partir de cette feuille de données d'extrémité de transformateur, représentant le courant magnétisant et les pertes dans le fer. Il convient de fournir toutes les valeurs du transformateur pour une info d'extrémité de transformateur uniquement.
[0..1]	[0..*] FromMeshImpedance	TransformerMeshImpedance	Toutes les impédances de maille entre cette extrémité de transformateur 'to' et les autres extrémités de transformateur 'from'.
[0..*]	[0..*] ToMeshImpedances	TransformerMeshImpedance	Toutes les impédances de maille entre cette extrémité de transformateur 'from' et les autres extrémités de transformateur 'to'.
[0..1]	[0..1] TransformerStarImpedance	TransformerStarImpedance	Impédance de transformateur en étoile calculée à partir de cette feuille de données d'extrémité de transformateur.
[0..1]	[0..*] EnergisedEndNoLoadTest	NoLoadTest	Toutes les mesures d'essai à vide pour lesquelles cette extrémité de transformateur a été mise sous tension.
[1..1]	[0..*] OpenEndOpenCircuitTests	OpenCircuitTest	Toutes les mesures d'essai de circuit ouvert pour lesquelles cette extrémité de transformateur n'a pas été mise sous tension.
[1..1]	[0..*] EnergisedEndOpenCircuitTests	OpenCircuitTest	Toutes les mesures d'essai de circuit ouvert pour lesquelles cette extrémité de transformateur a été excitée.
[1..*]	[0..*] GroundedEndShortCircuitTests	ShortCircuitTest	Toutes les mesures d'essai de court-circuit pour lesquelles cette extrémité de transformateur a été mise en court-circuit.
[1..1]	[0..*] EnergisedEndShortCircuitTests	ShortCircuitTest	Toutes les mesures d'essai de court-circuit pour lesquelles cette extrémité de transformateur a été mise sous tension.
[1..*]	[1..1] TransformerTankInfo	TransformerTankInfo	Données de cuve de transformateur dont cette description d'extrémité fait partie.
[0..1]	[0..*] PowerSystemResource	PowerSystemResource	hérité de: AssetInfo
[0..1]	[0..*] Assets	Asset	hérité de: AssetInfo
[0..1]	[0..1] AssetModel	AssetModel	hérité de: AssetInfo
[0..1]	[0..*] DiagramObjects	DiagramObject	hérité de: IdentifiedObject
[1..1]	[0..*] Names	Name	hérité de: IdentifiedObject

6.5.17 TransformerTankInfo

Ensemble de données de cuve de transformateur, issues d'une bibliothèque d'équipement.

Le Tableau 92 montre tous les attributs de TransformerTankInfo.

Tableau 92 – Attributs de AssetInfo::TransformerTankInfo

nom	type	description
aliasName	String	hérité de: IdentifiedObject
mRID	String	hérité de: IdentifiedObject
name	String	hérité de: IdentifiedObject

Le Tableau 93 montre toutes les extrémités d'associations de TransformerTankInfo avec les autres classes.

Tableau 93 – Extrémités d'associations de AssetInfo::TransformerTankInfo avec les autres classes

[mult à partir de]	[mult vers] nom	type	description
[1..*]	[1..1] PowerTransformerInfo	PowerTransformerInfo	Données relatives au transformateur de puissance dont la description de cette cuve fait partie.
[1..1]	[1..*] TransformerEndInfos	TransformerEndInfo	Données relatives à toutes les extrémités décrites par ces données de cuve de transformateur.
[0..1]	[0..*] PowerSystemResource	PowerSystemResource	hérité de: AssetInfo
[0..1]	[0..*] Assets	Asset	hérité de: AssetInfo
[0..1]	[0..1] AssetModel	AssetModel	hérité de: AssetInfo
[0..1]	[0..*] DiagramObjects	DiagramObject	hérité de: IdentifiedObject
[1..1]	[0..*] Names	Name	hérité de: IdentifiedObject

6.5.18 TransformerTest

Résultat des essais des extrémités de transformateur, tels que court-circuit, circuit ouvert (excitation) ou essai à vide.

Le Tableau 94 montre tous les attributs de TransformerTest.

Tableau 94 – Attributs de AssetInfo::TransformerTest

nom	type	description
basePower	ApparentPower	Puissance de base à laquelle les essais ont été réalisés, généralement égale aux valeurs nominales de l'une des extrémités de transformateur impliquées.
temperature	Temperature	Température à laquelle l'essai a été réalisé.
aliasName	String	hérité de: IdentifiedObject
mRID	String	hérité de: IdentifiedObject
name	String	hérité de: IdentifiedObject

Le Tableau 95 montre toutes les extrémités d'associations de TransformerTest avec les autres classes.

Tableau 95 – Extrémités d'associations de AssetInfo::TransformerTest avec les autres classes

[mult à partir de]	[mult vers] nom	type	description
[0..1]	[0..*] DiagramObjects	DiagramObject	hérité de: IdentifiedObject
[1..1]	[0..*] Names	Name	hérité de: IdentifiedObject

6.5.19 WireInfo

Données relatives aux fils qui peuvent être spécifiées par phase de segment de ligne ou pour le segment de ligne dans son ensemble lorsque toutes ses phases ont les mêmes caractéristiques de fil.

Le Tableau 96 montre tous les attributs de WireInfo.

Tableau 96 – Attributs de AssetInfo::WireInfo

nom	type	description
insulated	Boolean	True (c'est-à-dire vrai) si le conducteur est isolé.
insulationMaterial	WireInsulationKind	(si conducteur isolé) Matériau utilisé pour l'isolant.
insulationThickness	Length	(si conducteur isolé) Épaisseur de l'isolant.
material	WireMaterialKind	Matériau du conducteur.
sizeDescription	String	Décrit le calibre ou la section du fil (par exemple 4/0, #2, 336.5).
radius	Length	Rayon extérieur du fil.
strandCount	Integer	Nombre de brins dans le conducteur.
coreRadius	Length	(en présence de matériau différent pour l'âme) Rayon de l'âme centrale.
coreStrandCount	Integer	(si utilisé) Nombre de brins dans l'âme d'acier.
gmr	Length	Rayon moyen géométrique (en anglais *geometric mean radius*). Si le conducteur est remplacé par un tube clos et fin de rayon GMR, sa réactance est alors identique à celle du conducteur réel.
ratedCurrent	CurrentFlow	Intensité de courant maximal que le fil peut supporter dans des conditions thermiques énoncées.
rAC25	ResistancePerLength	Résistance CA linéique du conducteur à 25 °C.
rAC50	ResistancePerLength	Résistance CA linéique du conducteur à 50 °C.
rAC75	ResistancePerLength	Résistance CA linéique du conducteur à 75 °C.
rDC20	ResistancePerLength	Résistance CC linéique du conducteur à 20 °C.
aliasName	String	hérité de: IdentifiedObject
mRID	String	hérité de: IdentifiedObject
name	String	hérité de: IdentifiedObject

Le Tableau 97 montre toutes les extrémités d'associations de WireInfo avec les autres classes.

Tableau 97 – Extrémités d'associations de AssetInfo::WireInfo avec les autres classes

[mult à partir de]	[mult vers] nom	type	description
[0..*]	[0..*] Impedances	PerLengthImpedance	Toutes les impédances calculées à partir de cette feuille de données de fil.
[0..1]	[0..*] PowerSystemResource	PowerSystemResource	hérité de: AssetInfo
[0..1]	[0..*] Assets	Asset	hérité de: AssetInfo
[0..1]	[0..1] AssetModel	AssetModel	hérité de: AssetInfo
[0..1]	[0..*] DiagramObjects	DiagramObject	hérité de: IdentifiedObject
[1..1]	[0..*] Names	Name	hérité de: IdentifiedObject

6.5.20 Enumération WireInsulationKind

Sorte d'isolant de fil.

Le Tableau 98 montre tous les libellés de WireInsulationKind.

Tableau 98 – Libellés de AssetInfo::WireInsulationKind

libellé	description
asbestosAndVarnishedCambric	Isolant de fil en amiante et toile vernie.
butyl	Isolant de fil en butyle.
ethylenePropyleneRubber	Isolant de fil en caoutchouc éthylène-propylène.
highMolecularWeightPolyethylene	Isolant de fil en polyéthylène de masse moléculaire élevée.
treeResistantHighMolecularWeightPolyethylene	Isolant de fil en polyéthylène de masse moléculaire élevée résistant à larborescense.
lowCapacitanceRubber	Isolant de fil en caoutchouc de faible capacité.
oilPaper	Isolant de fil en papier huilé.
ozoneResistantRubber	Isolant de fil en caoutchouc résistant à l'ozone.
beltedPilc	Isolant de fil sous gaine de plomb isolé.
unbeltedPilc	Isolant de fil sans gaine de plomb isolé.
rubber	Isolant de fil en caoutchouc.
siliconRubber	Isolant de fil en caoutchouc – silicium.
varnishedCambricCloth	Isolant de fil en toile vernie.
varnishedDacronGlass	Isolant de fil en fibre de verre dacron verni.
crosslinkedPolyethylene	Isolant de fil en polyéthylène réticulé.
treeRetardantCrosslinkedPolyethylene	Isolant de fil en polyéthylène réticulé retardateur d'arborescense.
highPressureFluidFilled	Isolant de fil rempli de fluide à haute pression.
other	Autre sorte d'isolant de fil.

6.5.21 Enumération WireMaterialKind

Sorte de matériau de fil.

Le Tableau 99 montre tous les libellés de WireMaterialKind.

Tableau 99 – Libellés de AssetInfo::WireMaterialKind

libellé	description
copper	Fil en cuivre.
steel	Fil en acier.
aluminum	Fil en aluminium.
aluminumSteel	Fil en aluminium-acier.
acsr	Conducteur en aluminium renforcé d'acier.
aluminumAlloy	Fil en alliage d'aluminium.
aluminumAlloySteel	Fil en alliage d'aluminium acier.
aaac	Conducteur en alliage d'aluminium – armé ferraillé.
other	Autre matériau de fil.

6.5.22 WirePosition

Identification, espacement et configuration des fils d'un Conductor (c'est-à-dire conducteur) en fonction d'une structure.

Le Tableau 100 montre tous les attributs de WirePosition.

Tableau 100 – Attributs de AssetInfo::WirePosition

nom	type	description
phase	SinglePhaseKind	Désignation monophasé ou neutre pour le fil avec cette position.
xCoord	Displacement	Distance horizontale signée allant du fil à cette position au point de référence commun.
yCoord	Displacement	Distance verticale signée du fil à cette position: au-dessus du sol (valeur positive) ou sous le sol – profondeur d'enfouissement (valeur négative).
aliasName	String	hérité de: IdentifiedObject
mRID	String	hérité de: IdentifiedObject
name	String	hérité de: IdentifiedObject

Le Tableau 101 montre toutes les extrémités d'associations de WirePosition avec les autres classes.

Tableau 101 – Extrémités d'associations de AssetInfo::WirePosition avec les autres classes

[mult à partir de]	[mult vers] nom	type	description
[1..*]	[0..1] WireSpacingInfo	WireSpacingInfo	Données relatives à l'espacement entre fils dont relève cette position de fil.
[0..1]	[0..*] DiagramObjects	DiagramObject	hérité de: IdentifiedObject
[1..1]	[0..*] Names	Name	hérité de: IdentifiedObject

6.5.23 WireSpacingInfo

Données relatives à l'espacement entre fils qui associent plusieurs positions de fil au segment de ligne, et permet de calculer les impédances de segment de ligne. Le nombre des phases peut être dérivé du nombre de positions de fil associées dont la phase n'est pas neutre.

Le Tableau 102 montre tous les attributs de WireSpacingInfo.

Tableau 102 – Attributs de AssetInfo::WireSpacingInfo

nom	type	description
isCable	Boolean	Si true (vrai), ces données d'espacement décrivent un câble.
phaseWireCount	Integer	Nombre de sous-conducteurs de fil dans le faisceau symétrique (généralement entre 1 et 4).
phaseWireSpacing	Length	Distance entre sous-conducteurs de fil dans un faisceau symétrique.
usage	WireUsageKind	Usage des fils associés.
aliasName	String	hérité de: IdentifiedObject
mRID	String	hérité de: IdentifiedObject
name	String	hérité de: IdentifiedObject

Le Tableau 103 montre toutes les extrémités d'associations de WireSpacingInfo avec les autres classes.

Tableau 103 – Extrémités d'associations de AssetInfo::WireSpacingInfo avec les autres classes

[mult à partir de]	[mult vers] nom	type	description
[0..1]	[0..*] Impedances	PerLengthImpedance	Toutes les impédances calculées à partir de cette feuille de données d'espacement de fil.
[0..1]	[1..*] WirePositions	WirePosition	Toutes les positions des fils monophasés (phase ou neutre) formant le conducteur.
[0..1]	[0..*] PowerSystemResource	PowerSystemResource	hérité de: AssetInfo
[0..1]	[0..*] Assets	Asset	hérité de: AssetInfo
[0..1]	[0..1] AssetModel	AssetModel	hérité de: AssetInfo

[mult à partir de]	[mult vers] nom	type	description
[0..1]	[0..*] DiagramObjects	DiagramObject	hérité de: IdentifiedObject
[1..1]	[0..*] Names	Name	hérité de: IdentifiedObject

6.5.24 Enumération WireUsageKind

Sorte d'usage des fils.

Le Tableau 104 montre tous les libellés de WireUsageKind.

Tableau 104 – Libellés de AssetInfo::WireUsageKind

libellé	description
transport	Le fil est utilisé dans un réseau très haute tension ou haute tension.
distribution	Le fil est utilisé dans un réseau moyenne tension.
secondary	Le fil est utilisé dans un circuit basse tension.
other	Autre sorte d'usage de fil.

6.6 Paquetage Work

6.6.1 Généralités

Ce paquetage contient les classes noyau d'informations qui soutiennent les applications de gestion de travaux et de planification d'extension de réseau.

La Figure 38 montre le diagramme de classe WorkInheritance.

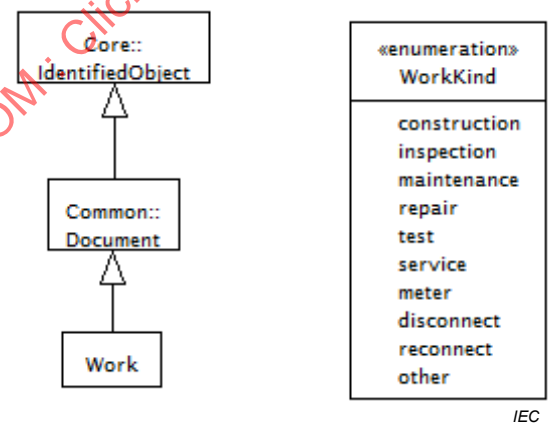


Figure 38 – Diagramme de classe Work::WorkInheritance

Ce diagramme montre la hiérarchie d'héritage pour les classes normatives issues de ce paquetage, ainsi que les énumérations et les types compound.

La Figure 39 montre le diagramme de classe WorkOverview.

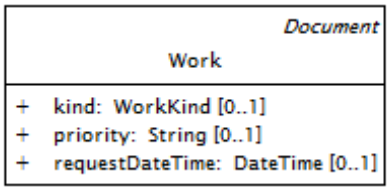


Figure 39 – Diagramme de classe Work::WorkOverview

Ce diagramme montre les classes normatives issues de ce paquetage.

6.6.2 Enumération WorkKind

Sorte des travaux.

Le Tableau 105 montre tous les libellés de WorkKind.

Tableau 105 – Libellés de Work::WorkKind

libellé	description
construction	Travaux de construction.
inspection	Travaux d'inspection.
maintenance	Travaux de maintenance.
repair	Travaux de réparation.
test	Travaux d'essais.
service	Travaux de service (entretien).
meter	Travaux de compteur.
disconnect	Travaux de débranchement.
reconnect	Travaux de rebranchement.
other	Autre sorte de travaux.

6.6.3 Work

Document utilisé pour demander, déclencher, suivre et enregistrer des travaux.

Le Tableau 106 montre tous les attributs de Work.

Tableau 106 – Attributs de Work::Work

nom	type	description
kind	WorkKind	Sorte des travaux.
priority	String	Priorité des travaux.
requestDateTime	DateTime	Date et heure à laquelle le travail est demandé.
type	String	hérité de: Document
createdDateTime	DateTime	hérité de: Document
lastModifiedDateTime	DateTime	hérité de: Document
revisionNumber	String	hérité de: Document
electronicAddress	ElectronicAddress	hérité de: Document
subject	String	hérité de: Document
title	String	hérité de: Document

nom	type	description
docStatus	Status	hérité de: Document
status	Status	hérité de: Document
aliasName	String	hérité de: IdentifiedObject
mRID	String	hérité de: IdentifiedObject
name	String	hérité de: IdentifiedObject

Le Tableau 107 montre toutes les extrémités d'associations de Work avec les autres classes.

Tableau 107 – Extrémités d'associations de Work::Work avec les autres classes

[mult à partir de]	[mult vers] nom	type	description
[0..*]	[0..*] Customers	Customer	Tous les clients pour lesquels ce travail est accompli.
[0..1]	[0..*] ConfigurationEvents	ConfigurationEvent	hérité de: Document
[0..1]	[0..*] DiagramObjects	DiagramObject	hérité de: IdentifiedObject
[1..1]	[0..*] Names	Name	hérité de: IdentifiedObject

6.7 Paquetage Customers

6.7.1 Généralités

Ce paquetage contient les classes d'informations centrales qui soutiennent les applications de facturation client.

La Figure 40 montre le diagramme de classe CustomersInheritance.

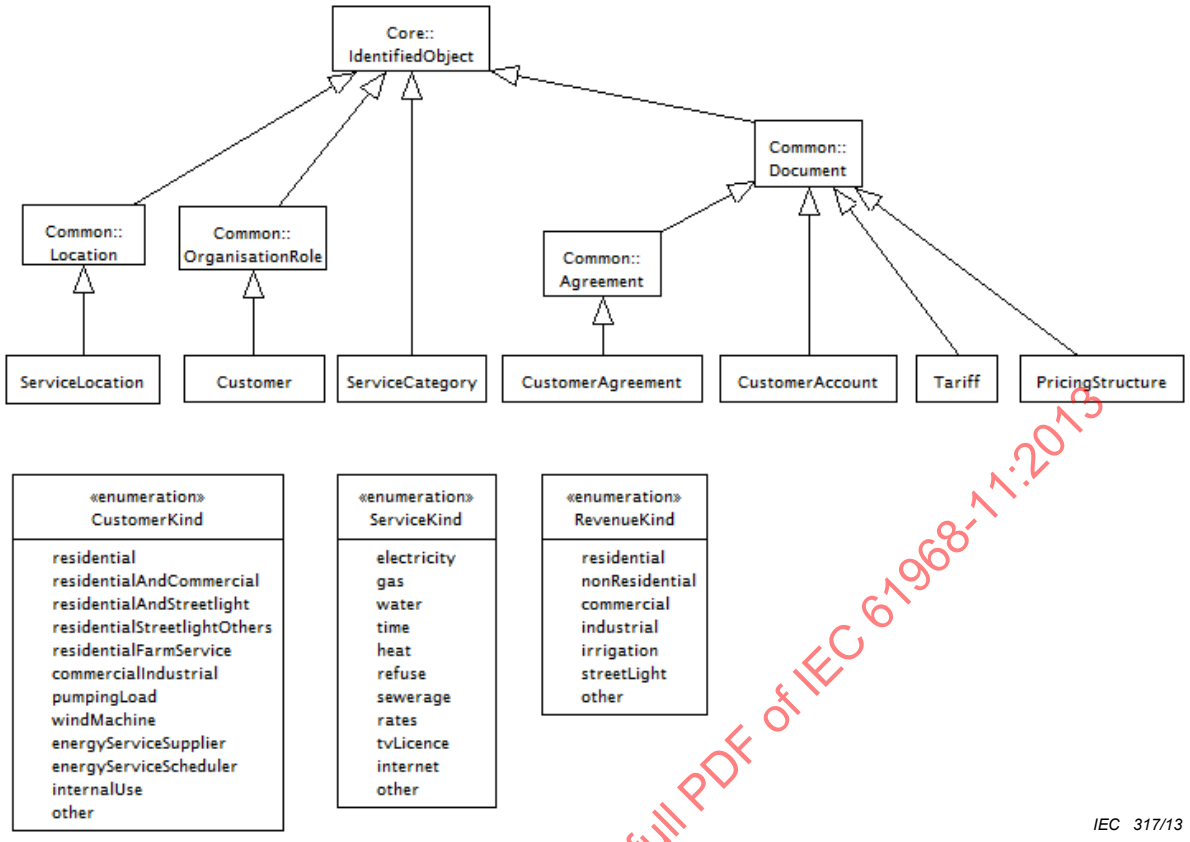


Figure 40 – Diagramme de classe Customers: :CustomersInheritance

Ce diagramme montre la hiérarchie d'héritage pour les classes normatives issues de ce paquetage, ainsi que les énumérations et les types compound.

La Figure 41 montre le diagramme de classe CustomersOverview.

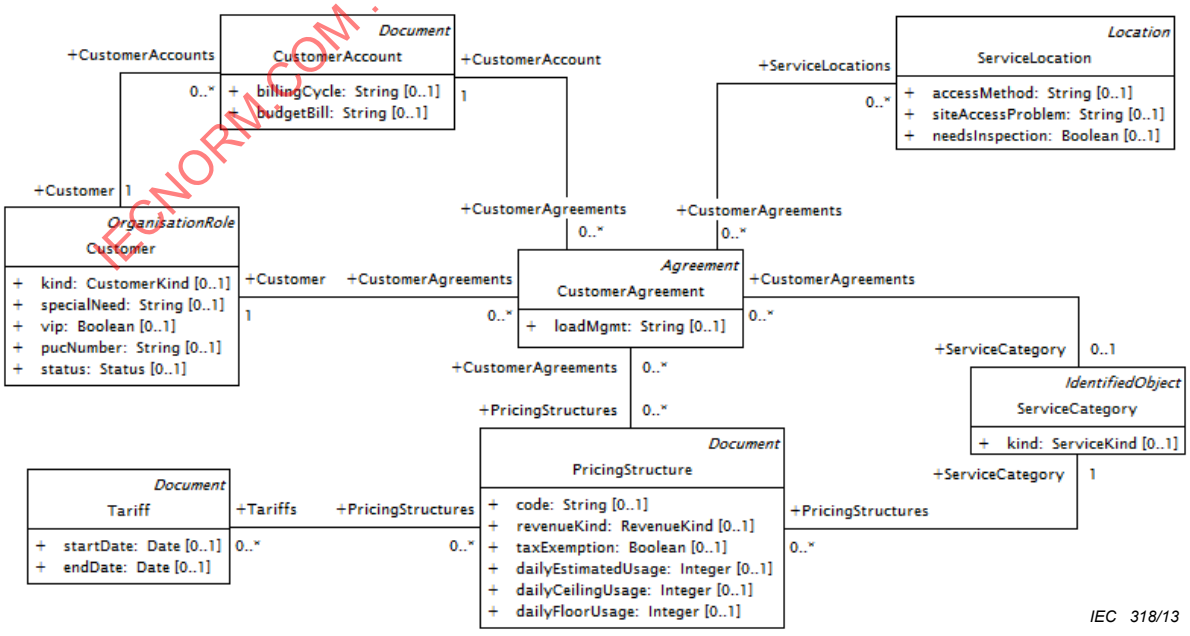


Figure 41 – Diagramme de classe Customers::CustomersOverview

Ce diagramme montre les classes normatives issues de ce paquetage.

6.7.2 Enumération CustomerKind

Sorte de client.

Le Tableau 108 montre tous les libellés de CustomerKind.

Tableau 108 – Libellés de Customers::CustomerKind

libellé	description
residential	Client résidentiel (d'habitation).
residentialAndCommercial	Client résidentiel et commercial.
residentialAndStreetlight	Client résidentiel et éclairage public.
residentialStreetlightOthers	Eclairage public résidentiel ou autre client associé.
residentialFarmService	Client pour service de ferme résidentielle.
commercialIndustrial	Client commercial – industriel.
pumpingLoad	Client pour charge de pompage.
windMachine	Client pour aérogénérateur.
energyServiceSupplier	Client en tant que fournisseur de service d'énergie.
energyServiceScheduler	Client en tant que planificateur de service d'énergie.
internalUse	Client pour utilisation interne.
other	Autre sorte de client.

6.7.3 Enumération RevenueKind

Classification comptable du type de recettes recueillies pour le CustomerAgreement, utilisée typiquement pour ventiler les comptes pour la comptabilité des recettes.

Le Tableau 109 montre tous les libellés de RevenueKind.

Tableau 109 – Libellés de Customers::RevenueKind

libellé	description
residential	Recettes résidentielles.
nonResidential	Recettes non résidentielles.
commercial	Recettes commerciales.
industrial	Recettes industrielles.
irrigation	Recettes d'irrigation.
streetLight	Recettes d'éclairage public.
other	Autre sorte de recettes.

6.7.4 Enumération ServiceKind

Sorte de service.

Le Tableau 110 montre tous les libellés de ServiceKind.

Tableau 110 – Libellés de Customers::ServiceKind

libellé	description
electricity	Service d'électricité.
gas	Service de gaz.
water	Service d'eau.
time	Service horaire.
heat	Service thermique.
refuse	Service de résidus urbains (déchets).
sewerage	Service d'assainissement (égouts).
rates	Services tarifaires (par exemple, impôt, taxe, péage, droit, tarif, etc.).
tvLicence	Service de redevance TV.
internet	Service Internet.
other	Autre sorte de service.

6.7.5 Customer

Organisation recevant des services de la part de ServiceSupplier.

Le Tableau 111 montre tous les attributs de Customer.

Tableau 111 – Attributs de Customers::Customer

nom	type	description
kind	CustomerKind	Sorte de client.
specialNeed	String	True (c'est-à-dire vrai) si l'organisation cliente a des besoins de services spéciaux tels que soutien vital, hôpitaux, etc.
vip	Boolean	True (vrai) s'il s'agit d'un client important. L'importance concerne des sujets différents de ceux contenus dans l'attribut 'specialNeed'.
pucNumber	String	(si applicable) Numéro d'identification de la Commission des entreprises de services publics (en anglais *PUC - Public utilities commission*).
status	Status	Statut de ce client.
aliasName	String	hérité de: IdentifiedObject
mRID	String	hérité de: IdentifiedObject
name	String	hérité de: IdentifiedObject

Le Tableau 112 montre toutes les extrémités d'associations de Customer avec les autres classes.

Tableau 112 – Extrémités d'associations de Customers::Customer avec les autres classes

[mult à partir de]	[mult vers] nom	type	description
[0..*]	[0..*] Works	Work	Toutes les tâches accomplies pour ce client.
[0..1]	[0..*] EndDevices	EndDevice	Tous les dispositifs finaux de ce client.
[1..1]	[0..*] CustomerAccounts	CustomerAccount	Tous les comptes de ce client.
[1..1]	[0..*] CustomerAgreements	CustomerAgreement	Tous les accords de ce client.
[0..1]	[0..*] ConfigurationEvents	ConfigurationEvent	hérité de: OrganisationRole
[0..*]	[0..1] Organisation	Organisation	hérité de: OrganisationRole
[0..1]	[0..*] DiagramObjects	DiagramObject	hérité de: IdentifiedObject
[1..1]	[0..*] Names	Name	hérité de: IdentifiedObject

6.7.6 CustomerAccount

Affectation d'un groupe de produits et de services achetés par le client par un accord client, utilisée comme mécanisme de facturation et paiement client. Il contient les informations communes issues de divers types d'accords pour créer des facturations (factures) pour un client et recevoir un paiement.

Le Tableau 113 montre tous les attributs de CustomerAccount.

Tableau 113 – Attributs de Customers::CustomerAccount

nom	type	description
billingCycle	String	Jour du cycle auquel le compte client associé sera normalement facturé, utilisé pour déterminer le moment de production de la facture.
budgetBill	String	Code de projet de budget.
type	String	hérité de: Document
createdDateTime	DateTime	hérité de: Document
lastModifiedDateTime	DateTime	hérité de: Document
revisionNumber	String	hérité de: Document
electronicAddress	ElectronicAddress	hérité de: Document
subject	String	hérité de: Document
title	String	hérité de: Document
docStatus	Status	hérité de: Document
status	Status	hérité de: Document
aliasName	String	hérité de: IdentifiedObject
mRID	String	hérité de: IdentifiedObject
name	String	hérité de: IdentifiedObject

Le Tableau 114 montre toutes les extrémités d'associations de CustomerAccount avec les autres classes.

Tableau 114 – Extrémités d'associations de Customers::CustomerAccount avec les autres classes

[mult à partir de]	[mult vers] nom	type	description
[0..1]	[0..*] PaymentTransactions	Transaction	Toutes les transactions de paiement pour ce compte client.
[1..1]	[0..*] CustomerAgreements	CustomerAgreement	Tous les accords pour ce compte client.
[0..*]	[1..1] Customer	Customer	Client titulaire de ce compte.
[0..1]	[0..*] ConfigurationEvents	ConfigurationEvent	hérité de: Document
[0..1]	[0..*] DiagramObjects	DiagramObject	hérité de: IdentifiedObject
[1..1]	[0..*] Names	Name	hérité de: IdentifiedObject

6.7.7 CustomerAgreement

Accord entre le client et le prestataire de services pour payer le service en un emplacement spécifique de service. Il consigne certaines informations de facturation concernant le type de service fourni en l'emplacement ServiceLocation et est utilisé pendant la création du débit pour déterminer le type de service.

Le Tableau 115 montre tous les attributs de CustomerAgreement.

Tableau 115 – Attributs de Customers::CustomerAgreement

nom	type	description
loadMgmt	String	Code de gestion de charge.
signDate	Date	hérité de: Agreement
validityInterval	DateTimeInterval	hérité de: Agreement
type	String	hérité de: Document
createdDateTime	DateTime	hérité de: Document
lastModifiedDateTime	DateTime	hérité de: Document
revisionNumber	String	hérité de: Document
electronicAddress	ElectronicAddress	hérité de: Document
subject	String	hérité de: Document
title	String	hérité de: Document
docStatus	Status	hérité de: Document
status	Status	hérité de: Document
aliasName	String	hérité de: IdentifiedObject
mRID	String	hérité de: IdentifiedObject
name	String	hérité de: IdentifiedObject

Le Tableau 116 montre toutes les extrémités d'associations de CustomerAgreement avec les autres classes.

Tableau 116 – Extrémités d'associations de Customers::CustomerAgreement avec les autres classes

[mult à partir de]	[mult vers] nom	type	description
[0..*]	[0..*] ServiceLocations	ServiceLocation	Tous les emplacements de service régis par cet accord client.
[0..1]	[0..*] AuxiliaryAgreements	AuxiliaryAgreement	Tous les accords auxiliaires (non liés à des services) qui se rapportent à cet accord client.
[0..*]	[0..*] PricingStructures	PricingStructure	Toutes les structures de tarification applicables à cet accord client.
[0..1]	[0..*] UsagePoints	UsagePoint	Tous les points de livraison de service régis par cet accord client.
[0..*]	[0..1] ServiceCategory	ServiceCategory	Catégorie de service pour cet accord.
[0..*]	[1..1] Customer	Customer	Client pour cet accord.
[0..*]	[1..1] CustomerAccount	CustomerAccount	Compte client propriétaire de cet accord.
[0..*]	[0..*] DemandResponsePrograms	DemandResponseProgram	Tous les programmes de réponse à la demande auxquels le client est inscrit par cet accord client.
[0..1]	[0..*] MeterReadings	MeterReading	(susceptible d'être déconseillé dans le futur) Tous les relevés de compteurs pour cet accord client.
[0..*]	[1..1] ServiceSupplier	ServiceSupplier	Fournisseur de service pour cet accord client.
[0..1]	[0..*] ConfigurationEvents	ConfigurationEvent	hérité de: Document
[0..1]	[0..*] DiagramObjects	DiagramObject	hérité de: IdentifiedObject
[1..1]	[0..*] Names	Name	hérité de: IdentifiedObject

6.7.8 PricingStructure

Regroupement de composantes de tarification et de prix utilisé dans la création des débits client et les critères d'éligibilité selon lesquels ces termes peuvent être proposés à un client. Les raisons du regroupement sont notamment les exigences relatives à l'état, à la classification des clients, aux caractéristiques de site, à la classification (c'est-à-dire, structure de tarification des droits, structure de tarification des dépôts, structure de tarification des services d'électricité, etc.) et à la comptabilité.

Le Tableau 117 montre tous les attributs de PricingStructure.

Tableau 117 – Attributs de Customers::PricingStructure

nom	type	description
code	String	Clé unique allouée à/par l'utilisateur pour cette structure de tarification, utilisée par les représentants de l'entreprise pour identifier la structure de tarification correcte en vue de l'allocation à un client. Pour les échelles tarifaires, il est souvent préfixé par un code d'état.
revenueKind	RevenueKind	(comptabilité) Sorte de recettes, souvent utilisée pour déterminer le délai de grâce accordé, avant que des actions de recouvrement ne soient lancées contre un client (les périodes de grâce varient entre les classes de recettes).

nom	type	description
taxExemption	Boolean	True (vrai) si la structure de tarification n'est pas imposable.
dailyEstimatedUsage	Integer	Utilisé en lieu et place de la moyenne estimée réellement calculée lorsque l'historique de l'usage n'est pas disponible, et saisi typiquement manuellement par la comptabilité des tiers.
dailyCeilingUsage	Integer	Quantité maximale absolue valide d'usage non exigé utilisée pour valider un usage non exigé facturé à un client.
dailyFloorUsage	Integer	Quantité minimale absolue valide d'usage non exigé utilisée pour valider un usage non exigé facturé à un client.
type	String	hérité de: Document
createdDateTime	DateTime	hérité de: Document
lastModifiedDateTime	DateTime	hérité de: Document
revisionNumber	String	hérité de: Document
electronicAddress	ElectronicAddress	hérité de: Document
subject	String	hérité de: Document
title	String	hérité de: Document
docStatus	Status	hérité de: Document
status	Status	hérité de: Document
aliasName	String	hérité de: IdentifiedObject
mRID	String	hérité de: IdentifiedObject
name	String	hérité de: IdentifiedObject

Le Tableau 118 montre toutes les extrémités d'associations de PricingStructure avec les autres classes.

Tableau 118 – Extrémités d'associations de Customers::PricingStructure avec les autres classes

[mult à partir de]	[mult vers] nom	type	description
[0..*]	[0..*] UsagePoints	UsagePoint	Tous les points de livraison de service (avec le compteur de prépaiement tournant comme un dispositif autonome, sans CustomerAgreement ou Customer) auxquels cette structure de tarification s'applique.
[0..1]	[0..*] Transactions	Transaction	Toutes les transactions appliquant cette structure de tarification.
[0..*]	[0..*] Tariffs	Tariff	Tous les tarifs utilisés par la structure de tarification.
[0..*]	[0..*] CustomerAgreements	CustomerAgreement	Tous les accords client avec la structure de tarification.
[0..*]	[1..1] ServiceCategory	ServiceCategory	Catégorie de service à laquelle cette structure de tarification s'applique.
[0..1]	[0..*] ConfigurationEvents	ConfigurationEvent	hérité de: Document
[0..1]	[0..*] DiagramObjects	DiagramObject	hérité de: IdentifiedObject
[1..1]	[0..*] Names	Name	hérité de: IdentifiedObject

6.7.9 ServiceCategory

Catégorie de service fourni au client.

Le Tableau 119 montre tous les attributs de ServiceCategory.

Tableau 119 – Attributs de Customers::ServiceCategory

nom	type	description
kind	ServiceKind	Sorte de service.
aliasName	String	hérité de: IdentifiedObject
mRID	String	hérité de: IdentifiedObject
name	String	hérité de: IdentifiedObject

Le Tableau 120 montre toutes les extrémités d'associations de ServiceCategory avec les autres classes.

Tableau 120 – Extrémités d'associations de Customers::ServiceCategory avec les autres classes

[mult à partir de]	[mult vers] nom	type	description
[0..1]	[0..*] ConfigurationEvents	ConfigurationEvent	Tous les événements de configuration créés pour cette catégorie de service.
[1..1]	[0..*] PricingStructures	PricingStructure	Toutes les structures de tarification applicables à cette catégorie de service.
[0..1]	[0..*] CustomerAgreements	CustomerAgreement	Tous les accords client dans cette catégorie de service.
[0..1]	[0..*] UsagePoints	UsagePoint	Tous les points d'usage qui délivrent cette catégorie de service.
[0..1]	[0..*] DiagramObjects	DiagramObject	hérité de: IdentifiedObject
[1..1]	[0..*] Names	Name	hérité de: IdentifiedObject

6.7.10 ServiceLocation

Emplacement immobilier communément appelé local.

Le Tableau 121 montre tous les attributs de ServiceLocation.

Tableau 121 – Attributs de Customers::ServiceLocation

nom	type	description
accessMethod	String	Méthode pour que la personne d'entretien accède aux emplacements de service appropriés. Par exemple, une description de l'endroit où obtenir une clé si l'installation n'est pas surveillée par du personnel et est sécurisée.
siteAccessProblem	String	Problèmes rencontrés antérieurement lors de la visite ou de l'accomplissement d'une tâche sur ce site. Les exemples comprennent: chient méchant, client violent, occupant injurieux, obstacles, situations dangereuses, etc.

nom	type	description
needsInspection	Boolean	True (vrai) si l'inspection est nécessaire pour les installations en cet emplacement de service. Elle pourrait être demandée par un client, en raison d'une violation suspecte, de problèmes environnementaux (par exemple, un incendie dans le voisinage) ou pour corriger des données incompatibles.
type	String	hérité de: Location
mainAddress	StreetAddress	hérité de: Location
secondaryAddress	StreetAddress	hérité de: Location
phone1	TelephoneNumber	hérité de: Location
phone2	TelephoneNumber	hérité de: Location
electronicAddress	ElectronicAddress	hérité de: Location
geoInfoReference	String	hérité de: Location
direction	String	hérité de: Location
status	Status	hérité de: Location
aliasName	String	hérité de: IdentifiedObject
mRID	String	hérité de: IdentifiedObject
name	String	hérité de: IdentifiedObject

Le Tableau 122 montre toutes les extrémités d'associations de ServiceLocation avec les autres classes.

Tableau 122 – Extrémités d'associations de Customers::ServiceLocation avec les autres classes

[mult à partir de]	[mult vers] nom	type	description
[0..1]	[0..*] EndDevices	EndDevice	Tous les dispositifs finaux qui mesurent le service livré à l'emplacement de service.
[0..1]	[0..*] UsagePoints	UsagePoint	Tous les points d'usage livrant le service (du même type) à cet emplacement de service.
[0..*]	[0..*] CustomerAgreements	CustomerAgreement	Tous les accords client régissant cet emplacement de service.
[0..1]	[0..*] PowerSystemResources	PowerSystemResource	hérité de: Location
[0..1]	[0..*] ConfigurationEvents	ConfigurationEvent	hérité de: Location
[0..*]	[0..1] CoordinateSystem	CoordinateSystem	hérité de: Location
[1..1]	[0..*] PositionPoints	PositionPoint	hérité de: Location
[0..1]	[0..*] Assets	Asset	hérité de: Location
[0..1]	[0..*] DiagramObjects	DiagramObject	hérité de: IdentifiedObject
[1..1]	[0..*] Names	Name	hérité de: IdentifiedObject

6.7.11 Tariff

Document, approuvé par l'agence de régulation responsable, énumérant les termes et conditions, y compris un recueil des prix, dans lesquels les services publics seront fournis. Il a un numéro unique dans l'état ou la province. Pour les échelles tarifaires, il est fréquemment alloué par la commission des services publics (en anglais *Public utilities commission*) affiliée.

Le Tableau 123 montre tous les attributs de Tariff.

Tableau 123 – Attributs de Customers::Tariff

nom	type	description
startDate	Date	Date à laquelle le tarif a été activé.
endDate	Date	(si le tarif devient inactif) Date à laquelle le tarif est terminé.
type	String	hérité de: Document
createdDateTime	DateTime	hérité de: Document
lastModifiedDateTime	DateTime	hérité de: Document
revisionNumber	String	hérité de: Document
electronicAddress	ElectronicAddress	hérité de: Document
subject	String	hérité de: Document
title	String	hérité de: Document
docStatus	Status	hérité de: Document
status	Status	hérité de: Document
aliasName	String	hérité de: IdentifiedObject
mRID	String	hérité de: IdentifiedObject
name	String	hérité de: IdentifiedObject

Le Tableau 124 montre toutes les extrémités d'associations de Tariff avec les autres classes.

Tableau 124 – Extrémités d'associations de Customers::Tariff avec les autres classes

[mult à partir de]	[mult vers] nom	type	description
[0..*]	[0..*] TariffProfiles	TariffProfile	Tous les profils tarifaires utilisant ce tarif.
[0..*]	[0..*] PricingStructures	PricingStructure	Toutes les structures de tarification utilisant ce tarif.
[0..1]	[0..*] ConfigurationEvents	ConfigurationEvent	hérité de: Document
[0..1]	[0..*] DiagramObjects	DiagramObject	hérité de: IdentifiedObject
[1..1]	[0..*] Names	Name	hérité de: IdentifiedObject

6.8 Paquetage Metering

6.8.1 Généralités

Ce paquetage contient les classes noyau d'informations qui soutiennent les applications de dispositifs finaux avec des classes spécialisées pour l'équipement de comptage et les locaux sont les dispositifs de réseau et les fonctions de relevé à distance. Ces classes sont généralement associées avec le point où un service est livré au client.

La Figure 42 montre le diagramme de classe MeteringInheritance.

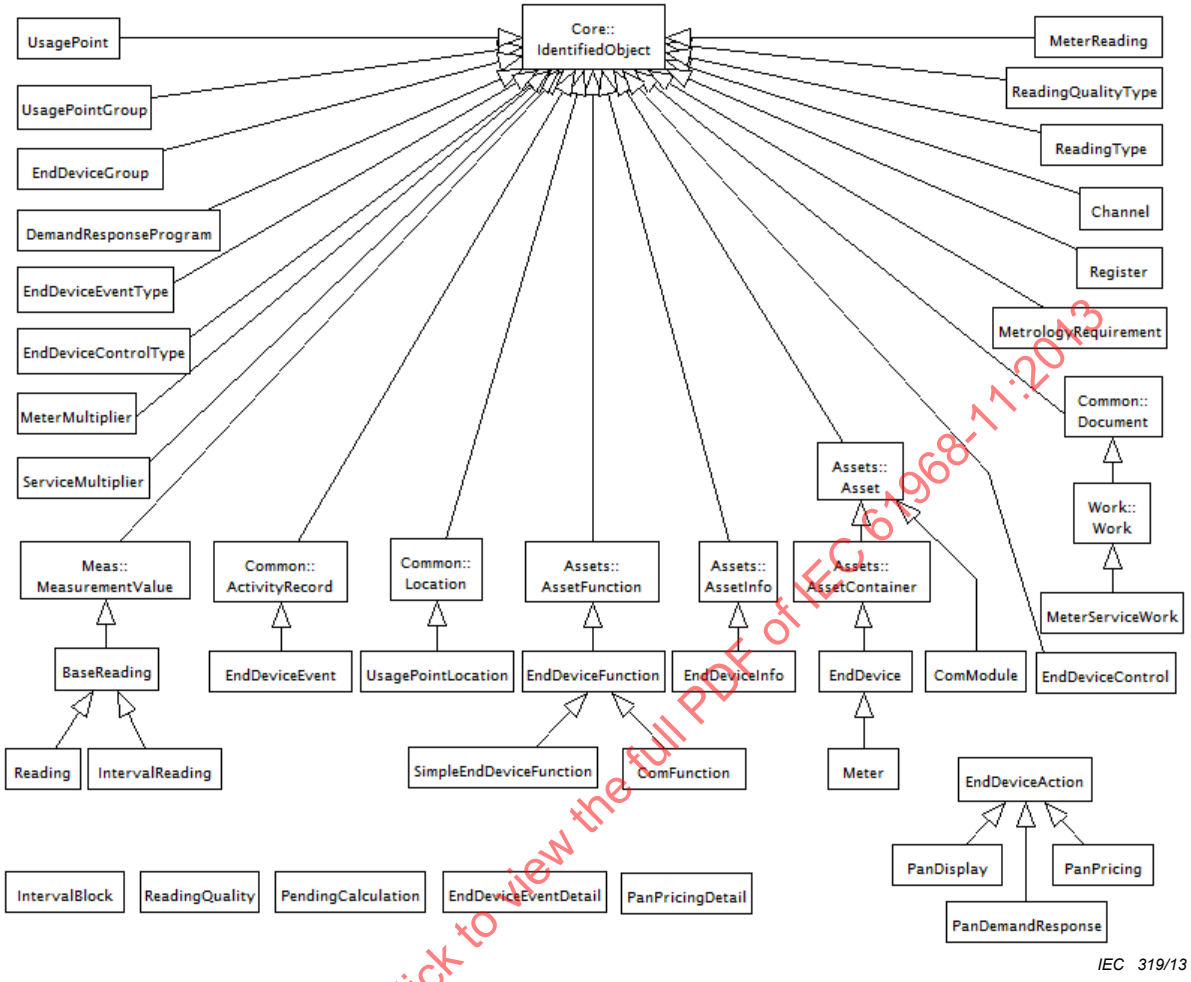


Figure 42 – Diagramme de classe Metering::MeteringInheritance

Ce diagramme montre la hiérarchie d'héritage pour les classes normatives issues de ce paquetage.