IEC 61968-11

Edition 1.0 2010-07

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 la distribution – Partie 11: Extensions du modèle d'information commun (CIM) pour la distribution

IEC Central Office
3, rue de Varembé
CH-1211 Geneva 20
Switzerland
Email: inmail@iec.ch
Web: 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.

- Catalogue of IEC publications: www.iec.ch/searchpub
The IEC on-line Catalogue enables you to search by a variety of criteria (reference number, text, technical committee,…). It also gives information on projects, withdrawn and replaced publications.
- IEC Just Published: www.iec.ch/online_news/justpub
Stay up to date on all new IEC publications. Just Published details twice a month all new publications released. Available on-line and also by email.
- Electropedia: www.electropedia.org
The world's leading online dictionary of electronic and electrical terms containing more than 20 000 terms and definitions in English and French, with equivalent terms in additional languages. Also known as the International Electrotechnical Vocabulary online.
- Customer Service Centre: www.iec.ch/webstore/custserv
If you wish to give us your feedback on this publication or need further assistance, please visit the Customer Service Centre FAQ or contact us:
Email: csc@iec.ch
Tel.: +41 22 919 02 11
Fax: +41 22 919 03 00

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.

- Catalogue des publications de la CEI: www.iec.ch/searchpub/cur_fut-f.htm
Le Catalogue en-ligne de la CEI vous permet d'effectuer des recherches en utilisant différents critères (numéro de référence, texte, comité d'études,…). Il donne aussi des informations sur les projets et les publications retirées ou remplacées.
- Just Published CEI: www.iec.ch/online_news/justpub
Restez informé sur les nouvelles publications de la CEI. Just Published détaille deux fois par mois les nouvelles publications parues. Disponible en-ligne et aussi par email.
- Electropedia: www.electropedia.org
Le premier dictionnaire en ligne au monde de termes électroniques et électriques. Il contient plus de 20 000 termes et définitions en anglais et en français, ainsi que les termes équivalents dans les langues additionnelles. Egalement appelé Vocabulaire Electrotechnique International en ligne.
- Service Clients: www.iec.ch/webstore/custserv/custserv_entry-f.htm
Si vous désirez nous donner des commentaires sur cette publication ou si vous avez des questions, visitez le FAQ du Service clients ou contactez-nous:
Email: csc@iec.ch
Tél.: +41 22 919 02 11
Fax: +41 22 919 03 00

IEC 61968-11

Edition 1.0 2010-07

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 la 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 978-2-88912-885-3

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.

This bilingual version (2013-01) corresponds to the monolingual English version, published in 2010-07.

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

FDIS	Report on voting
57/1058/FDIS	57/1073/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.

The French version of this standard has not been voted upon.

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

In informative sections of this document, words printed in Tahoma Bold 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.

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 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 International Standard 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 Parts 3 to 9 of IEC 61968, IEC 61968-13 and IEC 61968-14[1)]. The scope of this document 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 document, the distribution CIM (DCIM) model refers to the IEC TC 57 CIM model as defined by IEC 61970-301 and IEC 61968-11 (this document).

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 61970-302[2)] extends the CIM with packages that focus on Financial, Energy Scheduling, Reservation, and other market-related 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.

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

2) Under consideration.

2 Normative references

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

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*

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

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

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

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

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

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 markup language (XML) files

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

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

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 modeling notation

The CIM is defined using object-oriented modeling 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 TC 57 CIM packages

The comprehensive CIM is partitioned into groups of packages for convenience. These groups include:

- IEC 61970-301 (base CIM, defining data types and power system resources as required by typical EMS and DMS control centre applications);
- IEC 61968-11 (this document).

Figure 1 shows all currently defined TC 57 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.

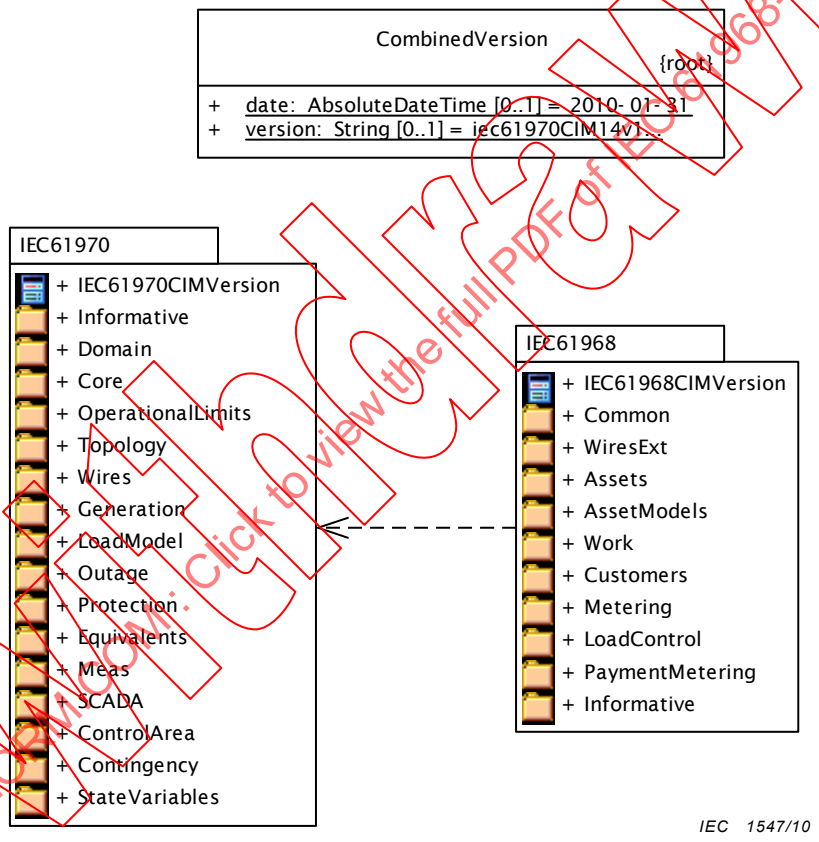


Figure 1 – TC 57 CIM packages

NOTE The contents of the base CIM referred to from within this specification were auto-generated from the base CIM UML electronic model release IEC61970CIM14v13.

4.2.3 CIM extensions for distribution packages (this document)

The base CIM model as defined by IEC 61970-301 defines a set of sub-packages which includes **Wires**, **Topology**, **Measurements**, **Equivalents** and **Core**, as well as several other ones. Parts 3 to 9 of IEC 61968, IEC 61968-13 and IEC 61968-14 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. 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.

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 IEC 61968 series, as well as different applications that use CIM for information exchange will typically define messages using classes from several packages, including some defined outside of this document.

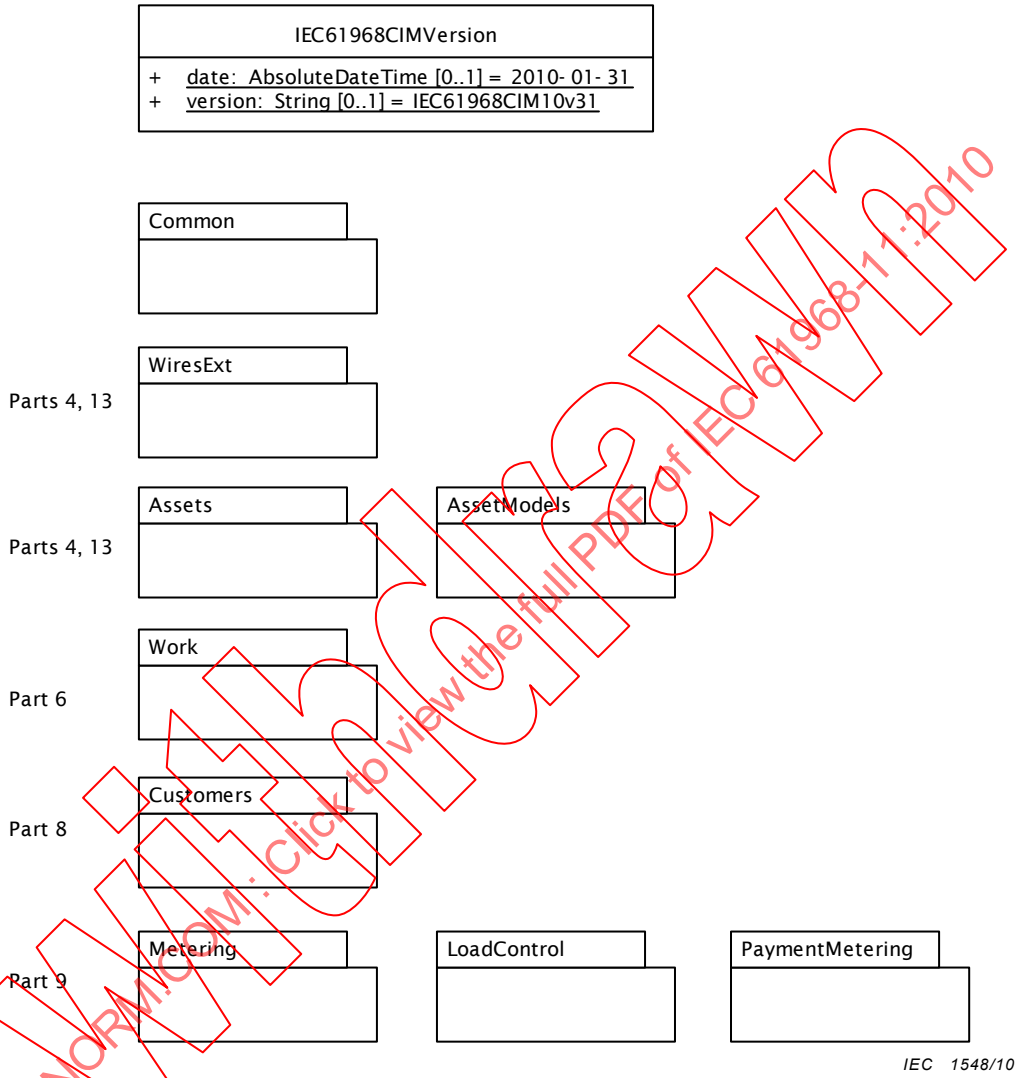


Figure 2 – CIM extensions for distribution (DCIM) top-level packages

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

NOTE The contents of the CIM defined in this specification were auto-generated from the CIM UML electronic model release IEC61968CIM10v31.

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 tools such as CIMTool to generate context specific messages in XSD, RDF, or OWL

format for Parts 3 to 9 of IEC 61968, IEC 61968-13 and IEC 61968-14. Other tool sets may also be used.

The purpose of this subclause (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 Subclause 4.3 of IEC 61970-301: CIM classes and relationships.

4.3.2 Scope of UML model

It is not the intent of this specification 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 Parts 3 to 9 of IEC 61968, IEC 61968-13 and IEC 61968-14. 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, but has never been published as an IEC standard. For this 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 61968-9 and IEC 61968-13. 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 Parts 3 to 9 of IEC 61968 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 11 for Edition 2 of this 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.

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.
- 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 (profile) definition

4.3.4.1 General

A profile is a 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 and information to exchange (specific to profile document IEC 61968-3 through IEC 61968-9 and IEC 61968-13).

One way of defining standard messages (profiles) is using applications such as the open source CIMTool. Other methods may be used, including hand coding.

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

4.3.4.2 Using XML schema

Messages defined within Parts 3 to 9 of IEC 61968 use combinations of nouns and verbs. The nouns usually refer to the classes defined within the DCIM model.

The verb usually indicates which attributes are required. In the case of **CREATE/CREATED** and **SHOW** 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 verbs defined for use within IEC 61968 compliant interfaces are specified in IEC 61968-1.

4.3.4.3 Using RDF schema

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 61970-552-4[3)], and is currently used by the profiles defined in IEC 61968-13.

4.4 DCIM model concepts and examples

4.4.1 General

This subclause (4.4) describes some examples of modeling in the distribution domain with DCIM. They are complementary to the base CIM examples presented in IEC 61970-301.

4.4.2 Key classes in DCIM

Base CIM in IEC 61970-301 mainly defines the function of electrical network elements through **PowerSystemResource** class and its subclasses, for the needs of information exchange in the context of control centre applications and systems. DCIM in this IEC 61968-11 document adds some key classes to support (a) physical description of those network elements, as well as (b) information exchange related to network operations and planning in the context of the whole utility enterprise.

Figure 3 shows some key classes in the DCIM.

3) Under consideration.

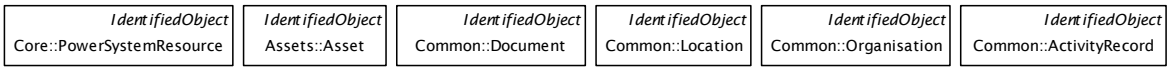


Figure 3 – DCIM key classes

An **Asset** is a tangible resource of the utility, including power system equipment, vehicles, tools, cabinets, buildings, etc. For electrical network equipment, the role of the asset is defined through the PowerSystemResource hierarchy, defined mainly in the wires model (refer to IEC 61970-301 and model package **IEC61970::Wires**, and model package **IEC61968::WiresExt** under Clause 6). Asset description places emphasis on the physical characteristics of the equipment fulfilling that role.

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.

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.

Organisations 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.

NOTE 1 Classes of the CIM are often used in many disparate contexts. For example, the same instance of organisation may be for a customer role in a context dealing with service requests and a manufacturer role in a different context dealing with the maintenance of a particular asset.

NOTE 2 Several of these key classes have relationship to the other classes (not shown in the figure), and in particular there are relationships among their subclasses. For example, an **Asset** plays a particular role in the electrical network as defined by the **PowerSystemResource** it is associated with. The **PowerSystemResource** may have a schematic location on a one-line diagram and a geographic location, whereas the **Asset** filling that role has a geographic location that a crew can service. If an **Asset** is removed from service, refurbished and then installed at another **Location**, history of the logical network and the physical assets may both be tracked by having instances of **ActivityRecord** associated with the **PowerSystemResource** as well as the **Asset**. The **PowerSystemResource** may be serving a particular **Organisation**, such as a supplier. This supplier is simultaneously a customer.

4.4.3 Single-phase and unbalanced loads

Figure 4 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.

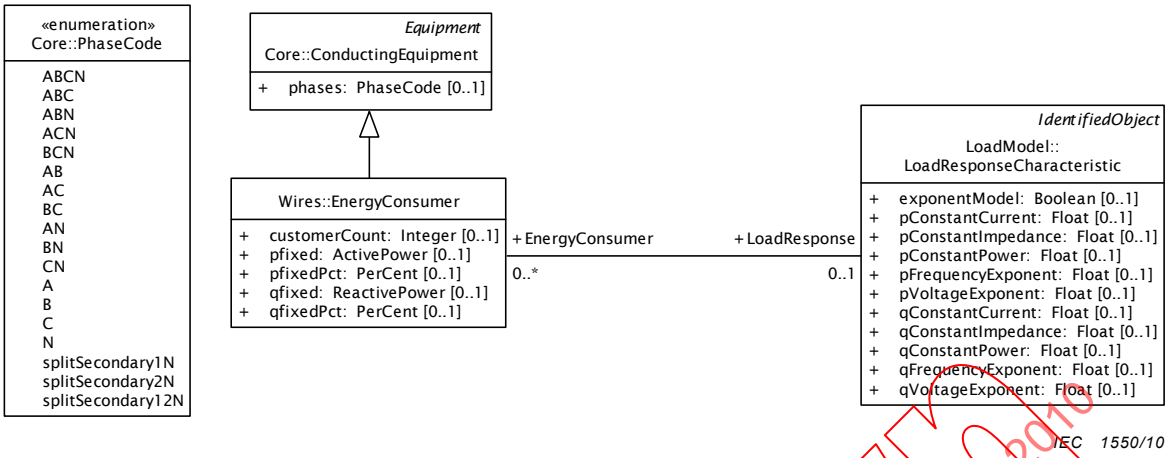


Figure 4 – DCIM load model

The **EnergyConsumer** class should be used to instantiate the load model, which can optionally be associated with a meter (through **ServiceDeliveryPoint**, not shown in the figure), but does not need to. **EnergyConsumer** inherits the **phases** attribute from **ConductingEquipment**, 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.

If a three-phase load is unbalanced, it can be modelled with three single-phase loads connected (through **Terminal**s) to the same **ConnectivityNode**. With single-phase or two-phase loads, the model should also include conductors or transformers that establish connectivity back to the source.

The load model could be some combination of constant current, constant power, or constant impedance. In that case, an instance of **LoadResponseCharacteristic** should be associated with the **EnergyConsumer** instance.

4.4.4 Distribution line segments

Figure 5 shows the classes available to model distribution line segments (i.e., conductors). There are four ways to describe the impedance parameters of a **DistributionLineSegment**, as a function of the kind of model exchange and the data available in the exporting system.

NOTE This edition of IEC 61968-11 (documenting DCIM10) and the corresponding edition of IEC 61970-301 (documenting base CIM14) reflect separate line segment models for distribution and transmission (T&D), respectively. The next editions should present a consolidated T&D line segment model.

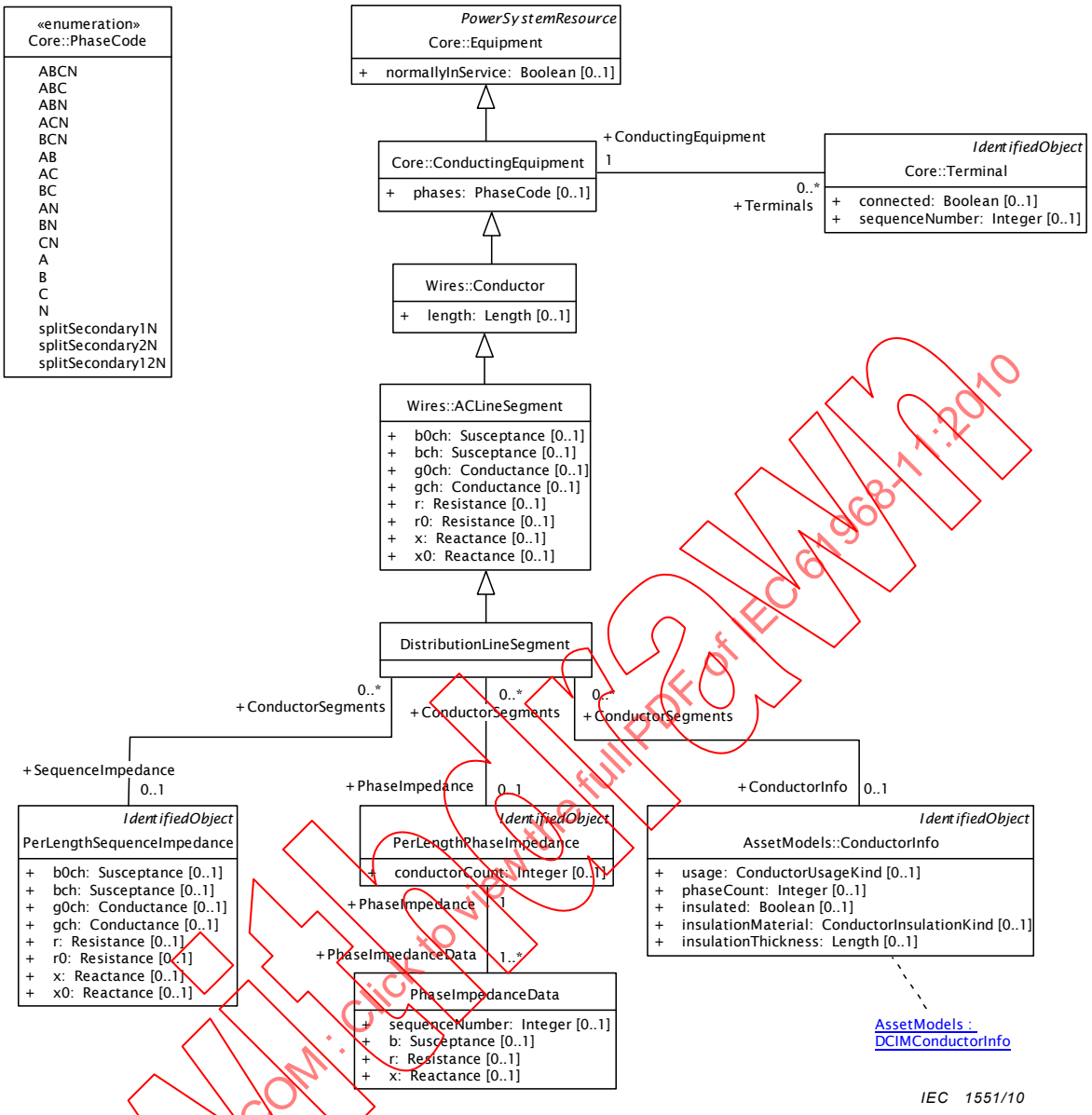


Figure 5 – DCIM line connectivity model

- Provide an association to **ConductorInfo**, detailed in Figure 6, for calculation of impedance and admittance per unit length from Carson's equations using physical wire and geometry data. **Conductor.length** attribute is required, because the instance electrical parameters are defined as (per-unit length parameters) * (**Conductor.length**). See also 4.4.4.3 below.

- Provide an association to **PerLengthPhaseImpedance**, which references pre-calculated NxN symmetric impedance and admittance matrices per unit length. The **conductorCount** (N) 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**). Because the **PhaseCode** enumeration is not ordered, the matrix rows and columns have to be in phase order. For **phases=ABC**, row 1 is always phase **A**, row 2 is phase **B**, and row 3 is phase **C**. This might describe a "line code" with phase **A** on the left, **B** in the middle, and **C** on the right. If the phasing is **BCA**, from left to right, that would require a different instance of **PerLengthPhaseImpedance** because the matrices will differ. **Conductor.length** is required for the **DistributionLineSegment**. See also 4.4.4.2 below.

- Provide an association to **PerLengthSequenceImpedance**. This class implements a library of "line codes" that have sequence impedances and line charging per unit length. **Conductor.length** is required. See 4.4.4.2 for usage of sequence parameters with 1-phase and 2-phase line segments.
- The inherited **ACLineSegment** attributes may be used if none of the preceding three associations is provided. **Conductor.length** is optional, and not used in the impedance calculation.

Figure 6 shows the **ConductorInfo** class hierarchy, used for defining line segment impedances from physical data (sometimes referred to as "line codes"). **ConductorInfo** is a base class that should not be instantiated directly.

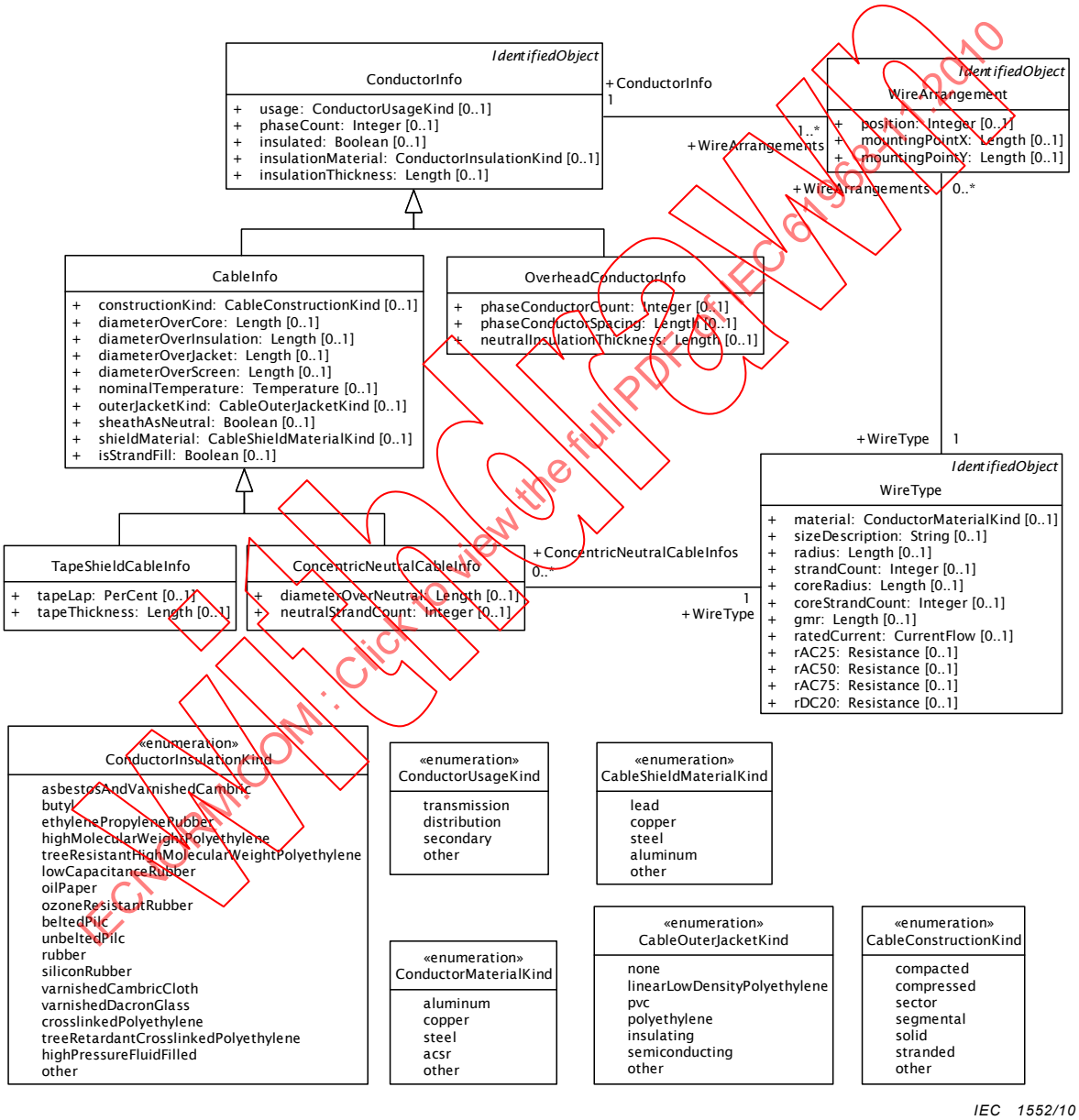


Figure 6 – DCIM conductor (line and cable datasheet) model

OverheadConductorInfo identifies the line physical data. Any conductors numbered above the **ConductorInfo.phaseCount** are assumed to be continuously grounded; the application may eliminate these conductors from the impedance and admittance matrices through Kron reduction. The **phaseConductorCount** and **phaseConductorSpacing** attributes of **OverheadConductorInfo** refer to sub-conductor bundling, which is not common for distribution lines, but may appear on high-voltage transmission lines in the model. **OverheadConductorInfo**

also allows for triplex secondary lines, by specifying the **insulated**, **insulationMaterial**, **insulationThickness**, and **neutralInsulationThickness** attributes.

WireType defines a library of wires. The 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.

WireArrangement defines the horizontal (**mountingPointX**) and vertical (**mountingPointY**) coordinates of each wire on the pole cross section. All three attributes are required. The wire height above ground is **mountingPointY**, including any sag effects. The wire horizontal position, **mountingPointX**, is measured from an arbitrary but consistent reference line. Common choices for the horizontal reference are the pole centerline, and the left-most wire position. The **WireArrangement** class mediates a many-to-many relationship between **ConductorInfo** and **WireType**. This model allows a different **WireType** for the neutral, and for each phase.

Underground cables will use **WireArrangement** and **WireType** in the same way as overhead lines. The two concrete classes that may be instantiated are **TapeShieldCableInfo** and **ConcentricNeutralCableInfo**. For concentric neutral cables, the copper neutral wires are defined by an association to **WireType**, with number and positioning determined by the **diameterOverNeutral** and **neutralStrandCount** attributes.

4.4.4.1 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 a **DistributionLineSegment** instance. The **bch**, **b0ch**, **gch**, and **g0ch** attributes are 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 thus must be 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 **DistributionLineSegment** 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 inherited **phases** attribute of **DistributionLineSegment** should be assigned an appropriate **PhaseCode** enumeration literal to show the phases actually present, such as **A**, **B**, **C**, **AB**, **BC**, **AC**, or **ABC**. 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.

4.4.4.2 Using phase impedances (unbalanced case)

Calculated matrix parameters may be transferred by referencing a **PerLengthPhaseImpedance** instance, in lieu of the physical model described in 4.4.4.3. 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 a **DistributionLineSegment** having phases **AB**, **AC**, or **BC**. Row 1 always corresponds to the first phase, which is **A**, **B**, or **C**, respectively. If the phase wires are physically transposed, such that **C** (for example) should be in row 1, a new instance of **PerLengthPhaseImpedance** is required.

4.4.4.3 Using physical parameters

For overhead lines, a physical model can be transferred through reference to an **OverheadConductorInfo** instance, which is associated with further classes shown in Figure 6. 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 **OverheadConductorInfo** instance, four **WireArrangement** instances that describe the four conductor positions, and two **WireType** instances describing the phase and neutral wire types. The **length** attribute of **Conductor** must be used, and many **DistributionLineSegment**s will typically refer to the same **OverheadConductorInfo** instance.

The **WireArrangement** instances link **WireType**s and **OverheadConductorInfo**s, in addition to defining the mounting points. These **WireArrangement**s must be sequenced in order to identify the phase wire assignments. The resistance attribute of **WireType** should be supplied for power frequency, and at the wire's desired operating temperature for calculations.

A single-phase, concentric neutral cable requires one instance of **ConcentricNeutralCableInfo**, one **WireArrangement** instance to specify the burial depth, and one **WireType** for the core conductor. A second **WireType** specifies the neutral strand; this is associated directly to **ConcentricNeutralCableInfo** rather than through a **WireArrangement**. A three-phase tape

shielded cable, with bare neutral conductor, would require one instance of **TapeShieldCableInfo** and four **WireArrangement**s for the three phases and neutral. There would also be two **WireType**s, one for the phase core conductor and another for the neutral.

4.4.5 Distribution transformers

4.4.5.1 Electrical model

Figure 7 shows the classes that model distribution transformer instances. They rely exclusively on a library of transformer types, detailed in Figure 8, to obtain most of the transformer parameters.

NOTE This edition of IEC 61968-11 (documenting DCIM10) and the corresponding edition of IEC 61970-301 (documenting base CIM14) reflect separate transformer models for distribution and transmission (T&D), respectively. The next editions should present a consolidated T&D transformer model.

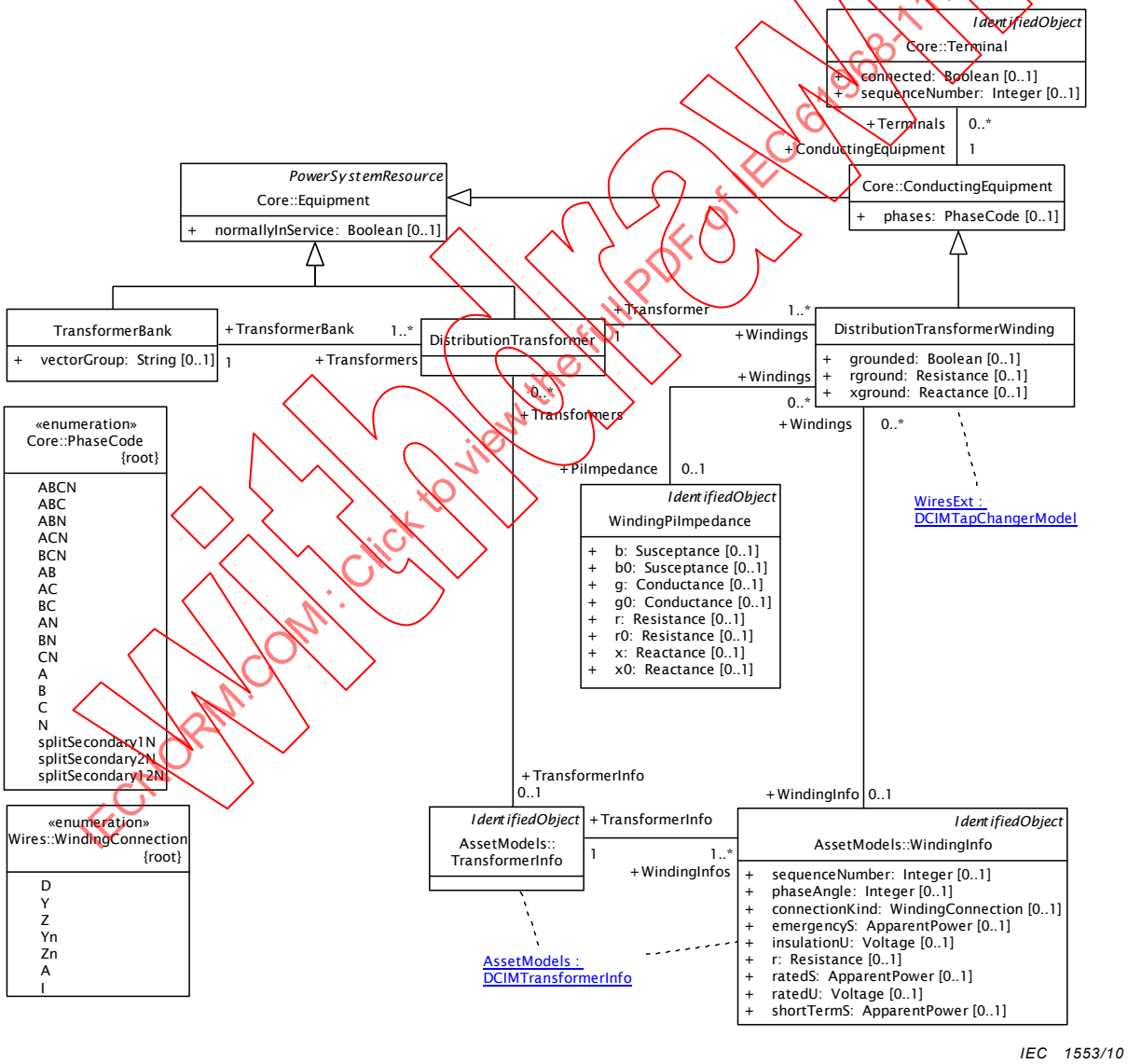


Figure 7 – DCIM transformer connectivity model

TransformerBank organizes the component three-phase and/or single-phase transformers. Some banks contain just one three-phase transformer, while others contain two or more single-phase transformers. 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. This class is equivalent to the **PowerTransformer** defined in IEC 61970-301, and it inherits from **Equipment**. Locations are often associated to the bank. The **vectorGroup** attribute for protective relaying is derived from the internal winding connections and phase angles; it uses standard nomenclature to describe any number of windings that may be included in the bank.

DistributionTransformer must reside in a transformer bank, and it organizes a flexible number of transformer windings. It references the **TransformerInfo** class in Figure 8 to obtain most of the data.

DistributionTransformerWinding is a **ConductingEquipment**, with phasing information and associated **Terminal**s. It references the **WindingInfo** class in Figure 8 to obtain data on the winding ratings. In addition to **ConductingEquipment.phases**, 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.

The **WindingPiImpedance** class allows for compatibility with IEC 61970-301 transformer models.

4.4.5.2 Physical model

Figure 8 shows the classes that allow for exchange of transformer physical data, sometimes referred to as “transformer codes” in applications.

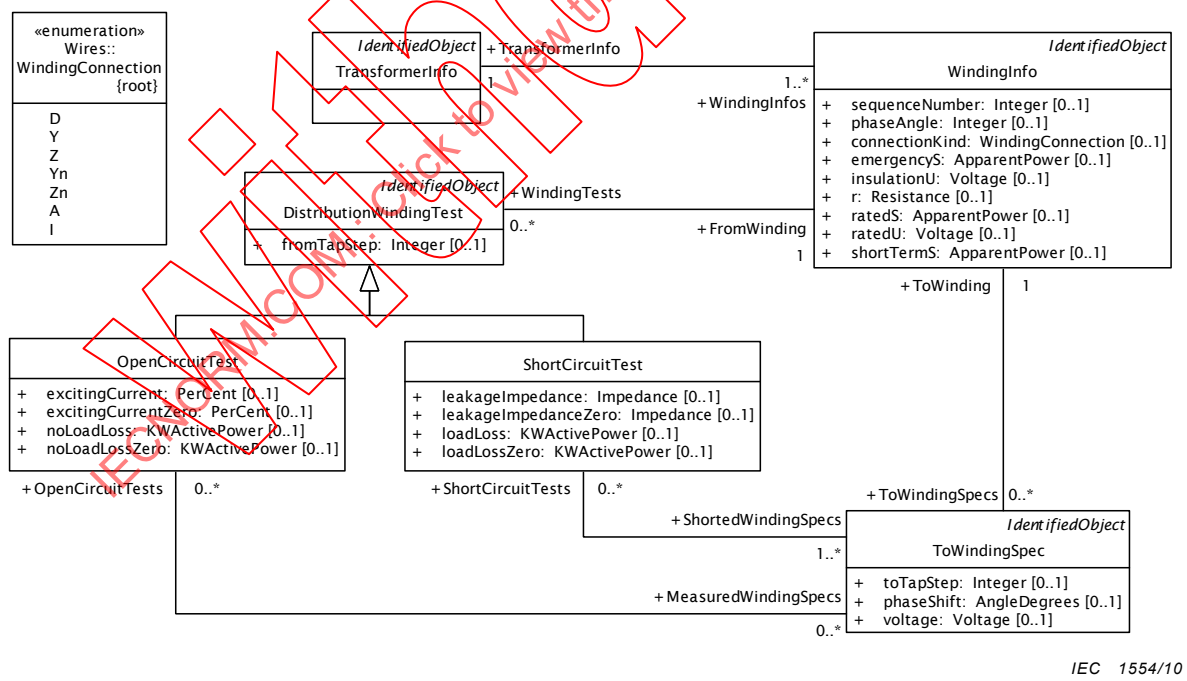


Figure 8 – DCIM transformer datasheet model

WindingConnection enumeration includes the standard **D**, **Y**, **Z**, **Yn**, and **Zn** nomenclature to describe delta, wye, zig-zag, 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.

TransformerInfo is referenced by the **DistributionTransformer**. It defines most of the transformer data. In the model at present, this class has no attributes of its own, and serves only to organize the winding attributes into a library. **TransformerInfo** collects associations to **WindingInfo**, such that the **WindingType** enumeration is not used. This generalizes to any number of windings.

WindingInfo includes all of the winding rating information, including **ratedU**, **ratedS**, **connectionKind**, **phaseAngle**, **shortTermS**, **emergencyS**, and **insulationU**. **r** is the winding's DC resistance. **sequenceNumber** is the winding's order in the **vectorGroup**; the high-voltage winding is always 1. The **sequenceNumber** increases as the **ratedU** decreases, but some transformers have two windings with the same **ratedU**, in which case the **sequenceNumber** distinguishes between them.

DistributionWindingTest with its attributes and subclasses covers short-circuit and open-circuit tests, for either single-phase or three-phase transformers, and in the case of three-phase transformers, either positive-sequence or zero-sequence excitation. For both open-circuit and short-circuit tests, only one winding ("from") is excited. The "from" winding is associated to the parent **DistributionWindingTest** class. For short-circuit tests, one or more other ("to") windings will be short-circuited. For open-circuit tests, there may be voltage and angle data on the other "to" windings.

OpenCircuitTest's attributes **excitingCurrent**, **excitingCurrentZero**, **noLoadLoss**, and **noLoadLossZero** are all measured on the excited ("from") winding. Positive and zero sequence test data can be reported in the same instance of **OpenCircuitTest**. This test is generally done with rated voltage applied to the excited winding. The test may include measurements of induced voltage and angle on other windings; these values would be reported in associated instances of **ToWindingSpec**.

ShortCircuitTest is done by circulating rated current through the "from" winding, with one or more "to" windings short circuited. Positive and zero sequence test data can be reported in the same instance of **ShortCircuitTest**. The AC resistances derived from **loadLoss** and **loadLossZero** are likely to differ from the winding DC resistances, **WindingInfo.r**, which are obtained from a separate test.

ToWindingSpec class defines the windings short circuited for a given **ShortCircuitTest,** and only the **toTapStep** attribute applies. For an **OpenCircuitTest**, the **voltage** and **phaseShift** attributes contain measured voltage and angle on the associated "to" winding. Instances of this class must be associated to either an **OpenCircuitTest** or a **ShortCircuitTest**, but not both.

4.4.5.3 Tap changer model

Figure 9 shows the classes used to model an unbalanced voltage regulator. A **DistributionTapChanger**, which inherits from **RatioTapChanger**, is associated to a **DistributionTransformerWinding**. Each regulator uses autonomous local control, so that **RegulationSchedule** and **TapSchedule** (present in IEC 61970-301 for power transformers) are not used. Some important attributes of those classes were copied to **DistributionTapChanger**. Phase angle regulators and variation curves are also not generally used on distribution systems.

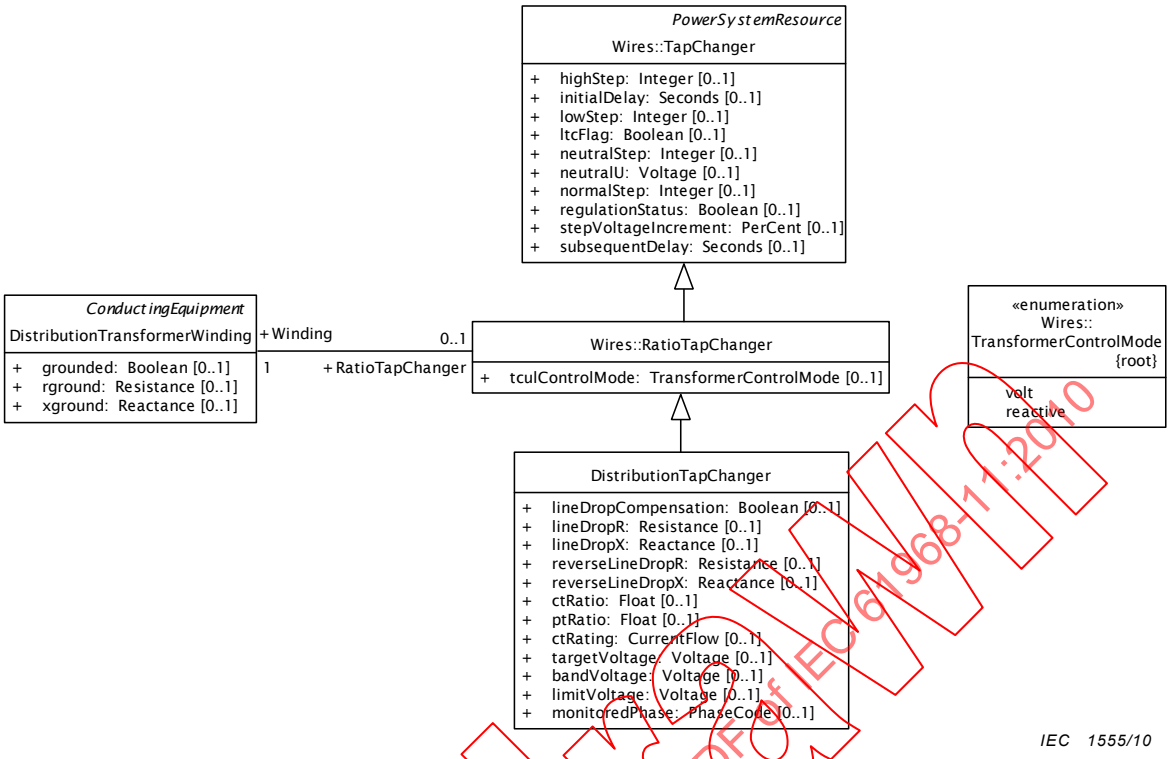


Figure 9 – DCIM tap changer model

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 **TransformerBank** containing three **DistributionTransformers**, and a total of six **DistributionTransformerWinding**s. There will be three instances of **DistributionTapChanger**, each associated to a different **DistributionTranformerWinding**. The **monitoredPhase** attribute of **DistributionTapChanger** should match the phase connection of its associated winding. 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. This could be modelled with a bank of one three-phase transformer and a total of two three-phase **DistributionTransformerWinding**s. There would be just one **DistributionTapChanger** associated to one winding. The **monitoredPhase** attribute of **DistributionTapChanger** 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 the regulator.

4.4.5.4 Example distribution transformer

The transformer in Figure 10 shows an open wye/open delta bank, which is used to supply inexpensive three-phase service to smaller customers.

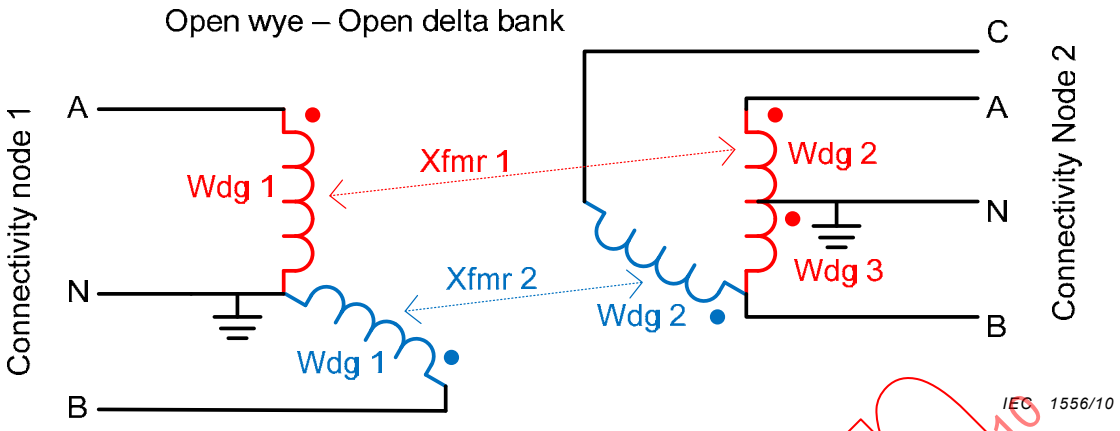


Figure 10 – Example of a distribution transformer that can be modelled with DCIM

Table 1 shows some of the important attribute values for this example.

Table 1 – Open wye/open delta transformer bank connections

Distribution Transformer	Distribution TransformerWinding	ratedU	ratedS	connectionType	phaseAngle	phaseCode
Xfmr 1	Wdg 1	7200	100e3	I	0	AN
	Wdg 2	120	50e3		0	AN
	Wdg 3	120	50e3	I	6	BN
Xfmr 2	Wdg 1	7200	50e3	I	0	BN
	Wdg 2	240	50e3	I	0	BC

A phase shift “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 from this transformer bank. Other connected equipment, such as a line segment, could add phase **C**. **ConnectivityNode** 2 will have phases **ABCN** present from this transformer bank. 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 **DistributionTransformerWinding**.

4.4.6 Connectivity with unbalanced phases

4.4.6.1 General

The distribution model connectivity follows the CIM connectivity model, i.e., it relies on **ConductingEquipment**, **Terminal**, and **ConnectivityNode** classes (refer to IEC 61970-301, Subclause 4.4.2 Connectivity model). **ConductingEquipment** has a **phase** attribute to describe the phase information of conducting equipment. **ConnectivityNode**s are points where **Terminal**s of **ConductingEquipment** are connected together with zero impedance. It does not have phase information. Relationship between a **ConnectivityNode** and its container is modelled through **ConnectivityNodeContainer** class.

NOTE The presented CIM connectivity and phasing model supports the exchange of an as-built network model. However, in distribution network operations there are cases where a **ConductingEquipment** has different state for each **phase**. For example, a three phase switch might be operated differently on each phase so that phase **A** is electrically disconnected and phases **BC** are electrically connected; as a result, the line that is connected to the switch may have different electrical state on each phase as well. Currently the CIM measurement model can not represent such cases since no phase information is provided on **Measurement**. That capability is much needed for distribution operation and control and may be present in the next edition of this IEC 61968-11 in support of IEC 61968-3.

For an extensive example of balanced three-phase sample network, refer to IEC 61970-301, Subclause 4.4.2.1 Connectivity and containment example. The examples provided in the next subclause illustrate the usage of connectivity model for unbalanced phase cases.

4.4.6.2 Connectivity examples for unbalanced phases

Generally speaking, a **ConnectivityNode** is created as the point where two or more devices are connected together. The devices, which are modelled as subtypes of **ConductingEquipment**, carry the phasing information through the **phase** attribute of **ConductingEquipment**.

Consider as example **ACLineSegment** or its subtype **DistributionLineSegment**, which are used to model electrical connectivity and characteristics (impedances) of overhead lines and cables. The physical line or cable needs to be electrically modelled in such a way that an **ACLineSegment** or **DistributionLineSegment** have one and only one **phase** attribute.

Figure 11 illustrates three examples with different phase configurations that are common in distribution networks.

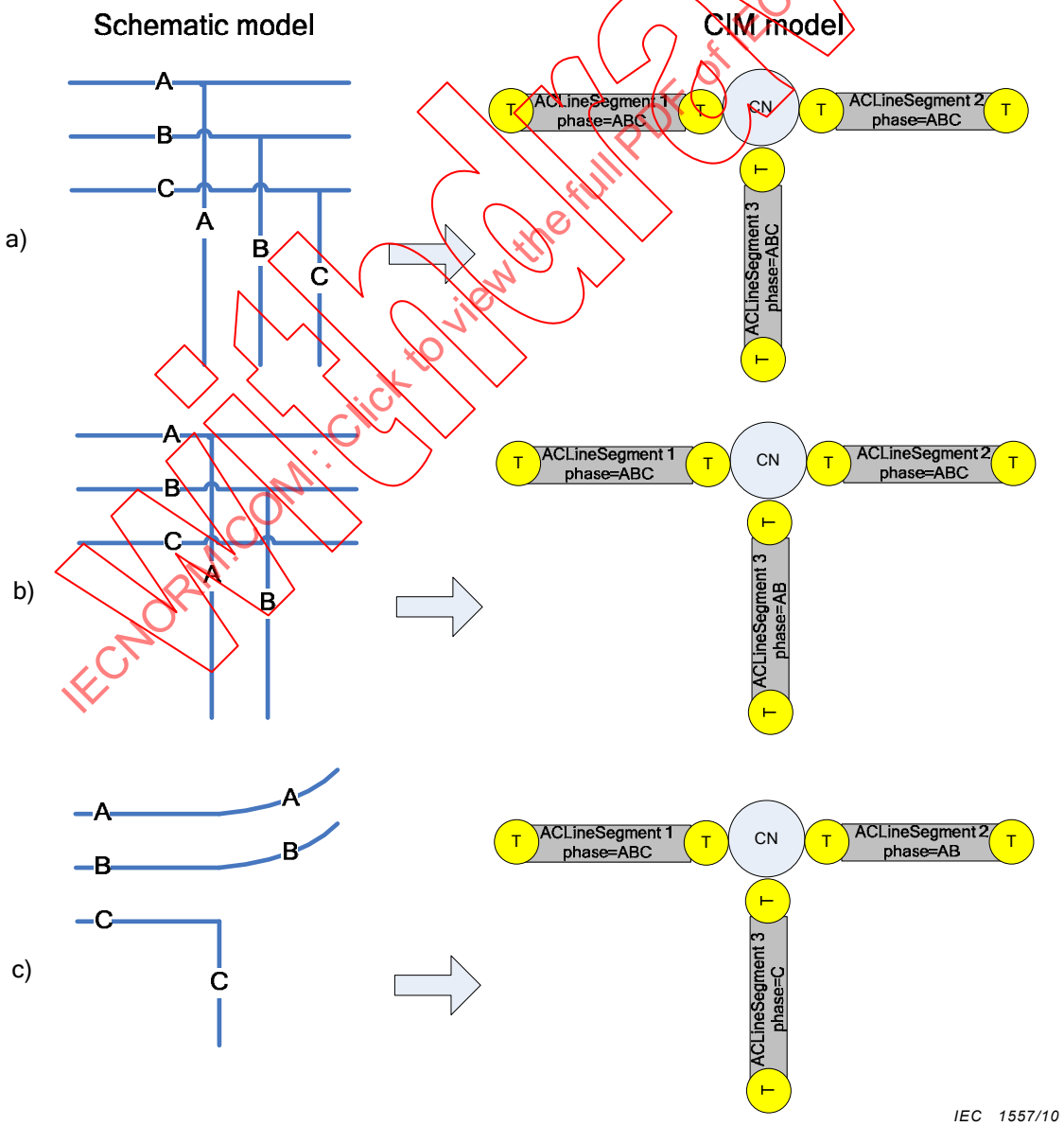


Figure 11 – DCIM connectivity for two-Terminal devices

Similar connectivity modelling is to be applied for any other CIM two-**Terminal** devices such as series shunts and switches.

Figure 12 illustrates two examples for modelling loads: balanced and unbalanced.

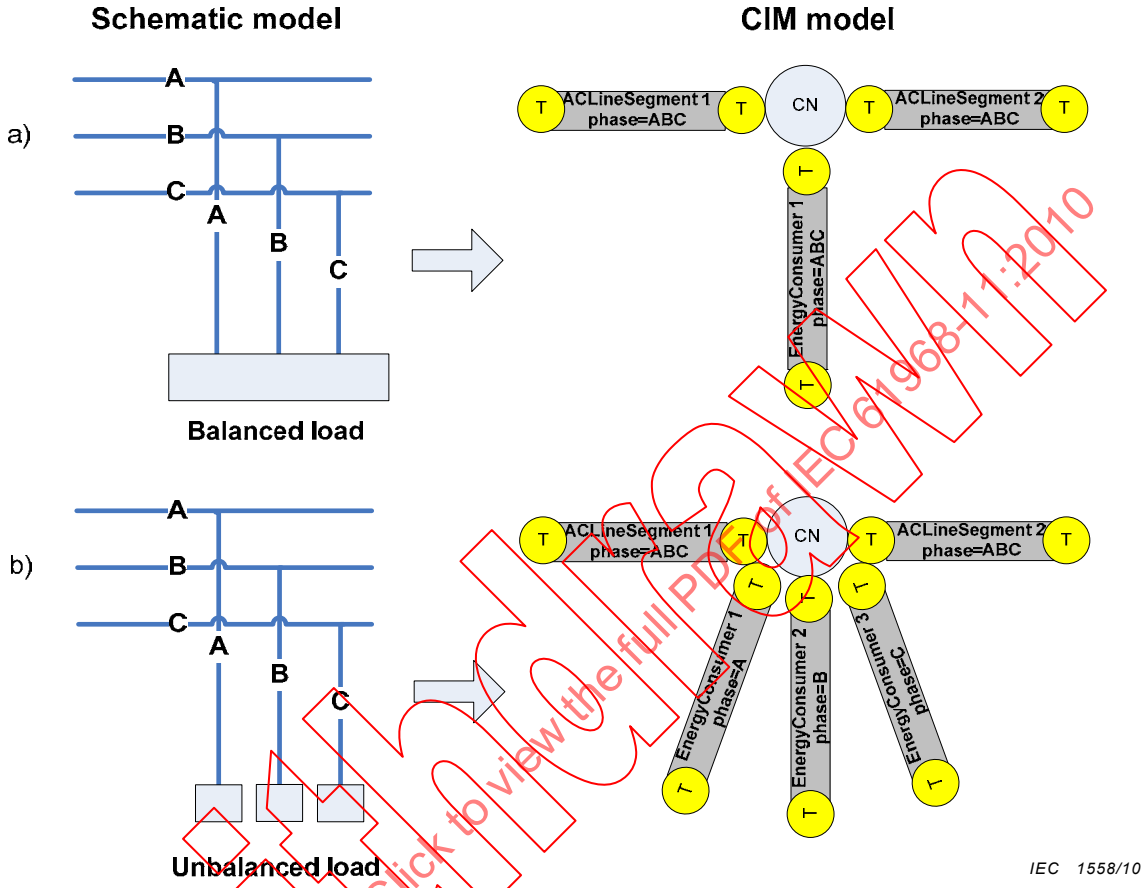


Figure 12 – DCIM connectivity for single-Terminal devices

Similar connectivity modelling is to be applied for any other CIM single-**Terminal** devices such as shunts, generators and transformer windings.

NOTE This edition of IEC 61968-11 does not support multi-state switches (modelled by **CompositeSwitch**) and does not provide any grouping concept for a switch bank (where phases can have different status), or for loads (with potentially different load characteristics).

4.4.7 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. The physical representation may also be essential in deriving attributes for the electrical representation, even if no explicit **Asset** class is used. Subclause 4.4.4 explains how the asset model can help calculate the electrical characteristic of a distribution line. The relationship between the two aspects also provides key information for extension planning, work management and outage management.

Figure 13 illustrates how **Asset** and **PowerSystemResource** are related in CIM for the purpose of some data exchange scenarios.

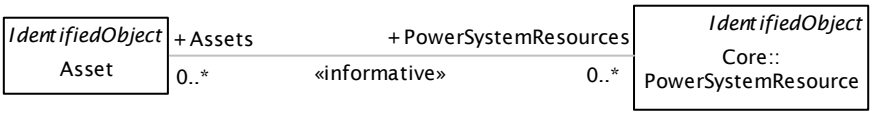


Figure 13 – DCIM assets and relation to power system resources

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 ratings, energisation status) are always modelled through **PowerSystemResource** and its subclasses, while physical aspects (such as data sheet ratings, dimensions, asset lifecycle) are always modelled through **Asset** and its related classes.

NOTE This edition of IEC 61968-11 contains **Asset** and **AssetModel** classes because they are used in the context of metering (IEC 61968-9), but not in the context of distribution network model exchange (IEC 61968-13). The limited set of assets-related classes in this edition contains those classes that support conductor and transformer modelling only, and the relationship **Asset**-**PowerSystemResource** is not used, thus tagged as informative. The next edition of this IEC 61968-11 should contain more complete asset model for the needs of distribution network model exchanges.

4.4.8 Locations and graphical representations

To allow for locating different entities in space, DCIM provides the class **Location**. Entities can be located by their different kinds of addresses as well as by position coordinates, **PositionPoint** in a given **CoordinateSystem**.

Location class is void of any graphical data exchange, although its **PositionPoint**s can be used to exchange coordinates that may be used by graphical applications.

Figure 14 shows how **Location** class can be used in relation with **Assets and PowerSystemResources**.

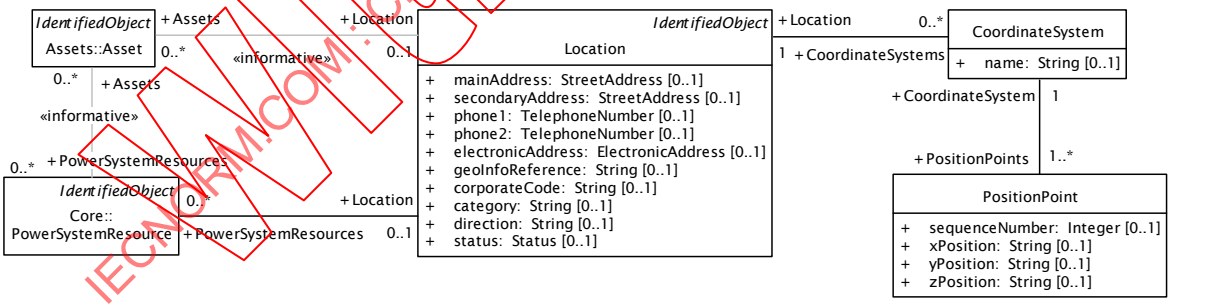


Figure 14 – DCIM power system resource and asset locations

NOTE Support for single line diagram exchanges, used in operations of transmission and some distribution control centres, is planned to be supported in base CIM, IEC 61970-301, and may be referred from a future edition of this IEC 61968-11. Support for geo-spatial data exchanges is out of the scope of this IEC 61968-11, since there are widely used and adopted standards for other domains that can be applied to electrical networks as well.

4.4.9 Metering

The **Metering** package introduces classes, shown in Figure 15 that are needed for the enterprise integration of metering systems and the information they exchange with other enterprise systems. A logical model for a meter is provided by **MeterAsset**, which inherits from **EndDeviceAsset**. **EndDeviceAssets** can detect and report **EndDeviceEvents**, report **MeterReadings** and accept **EndDeviceControls**. Within the model for meter readings, each

reading has a specific type and can be associated with an interval. **EndDeviceAsset** communication can be facilitated by individual addresses or group addresses. **EndDeviceAssets** are deployed at **ServiceDeliveryPoints**, providing the means to relate to locations and customers. A **MeterServiceWork** class is introduced to facilitate work related to the installation, maintenance and changeout of meters.

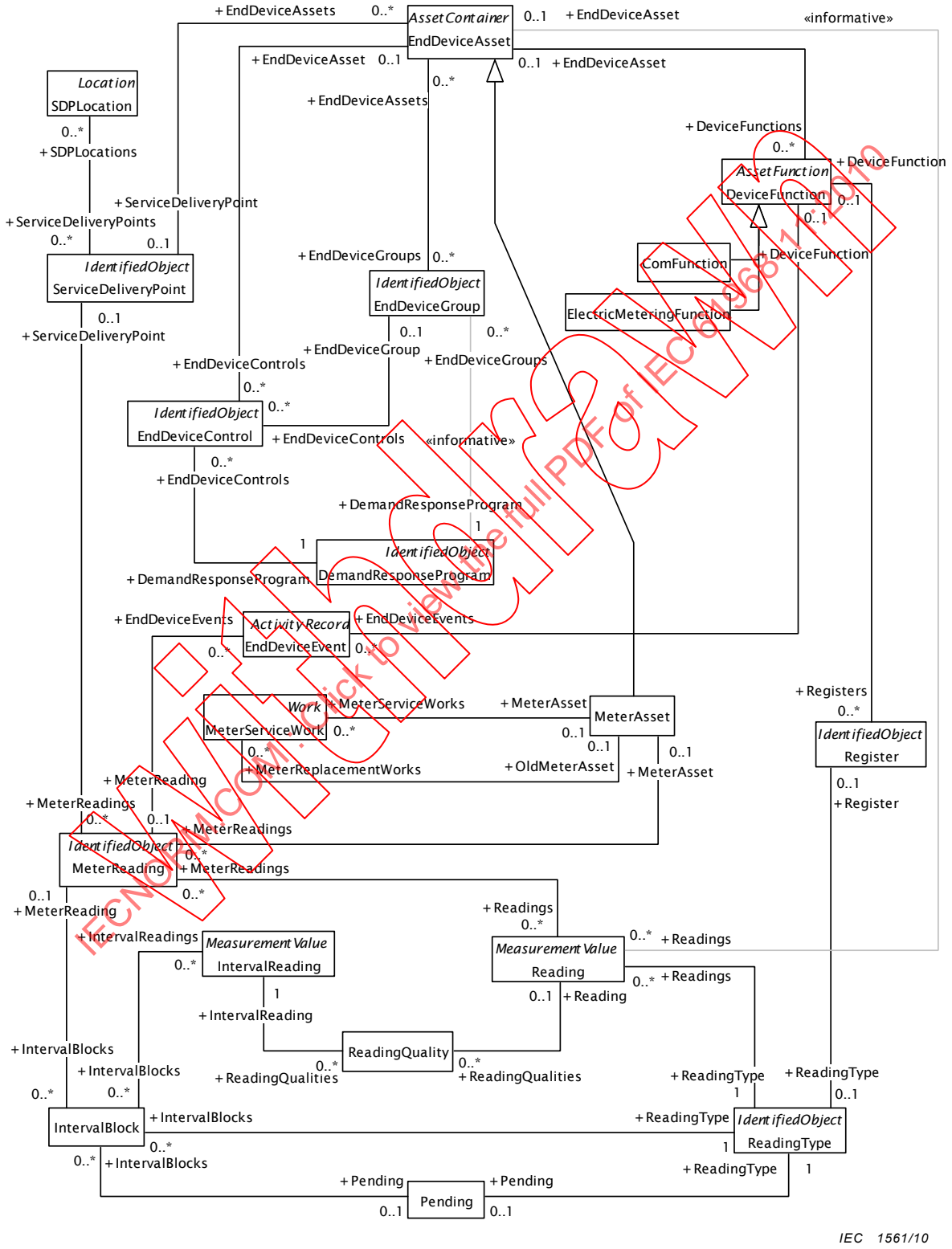


Figure 15 – DCIM metering model

4.4.10 PaymentMetering

4.4.10.1 Transacting

A payment metering system generally facilitates financial transactions between a customer and a service provider. The relevant information describing these transactions are 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 CIM classes found in the **PaymentMetering** package is shown in Figure 16.

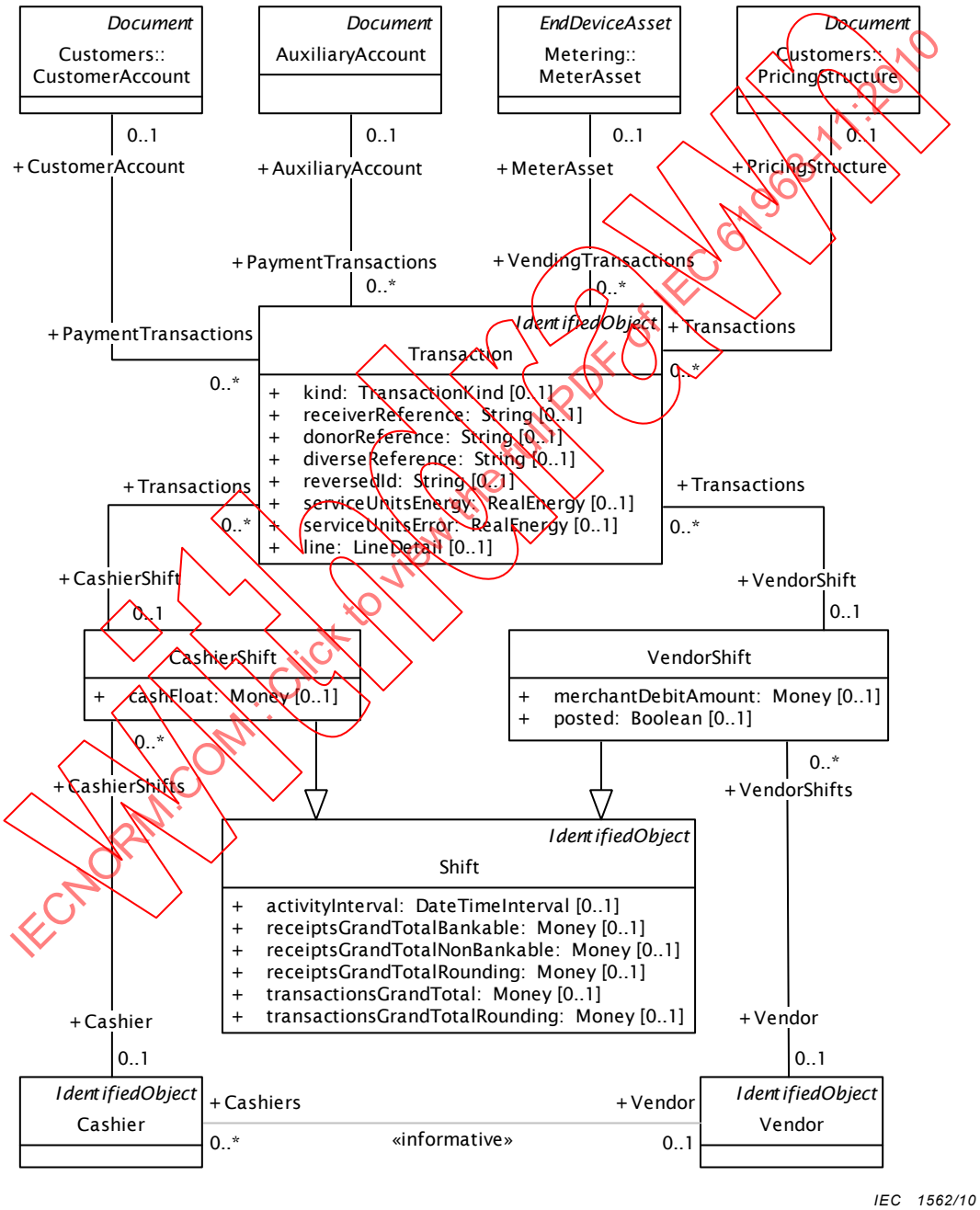


Figure 16 – DCIM transacting model

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**.

4.4.10.2 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 17.

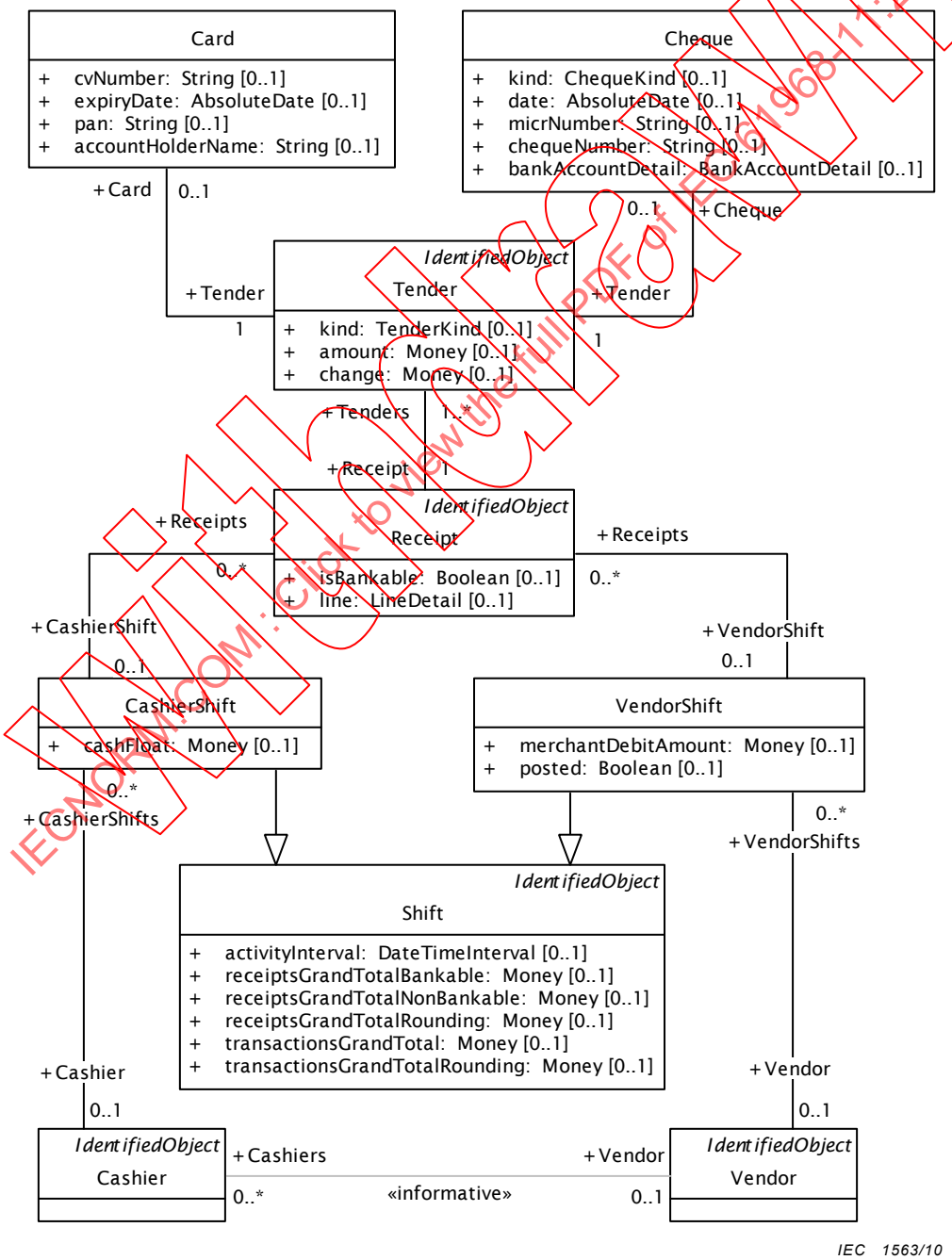


Figure 17 – DCIM receipting model

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**.

4.4.10.3 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 18.

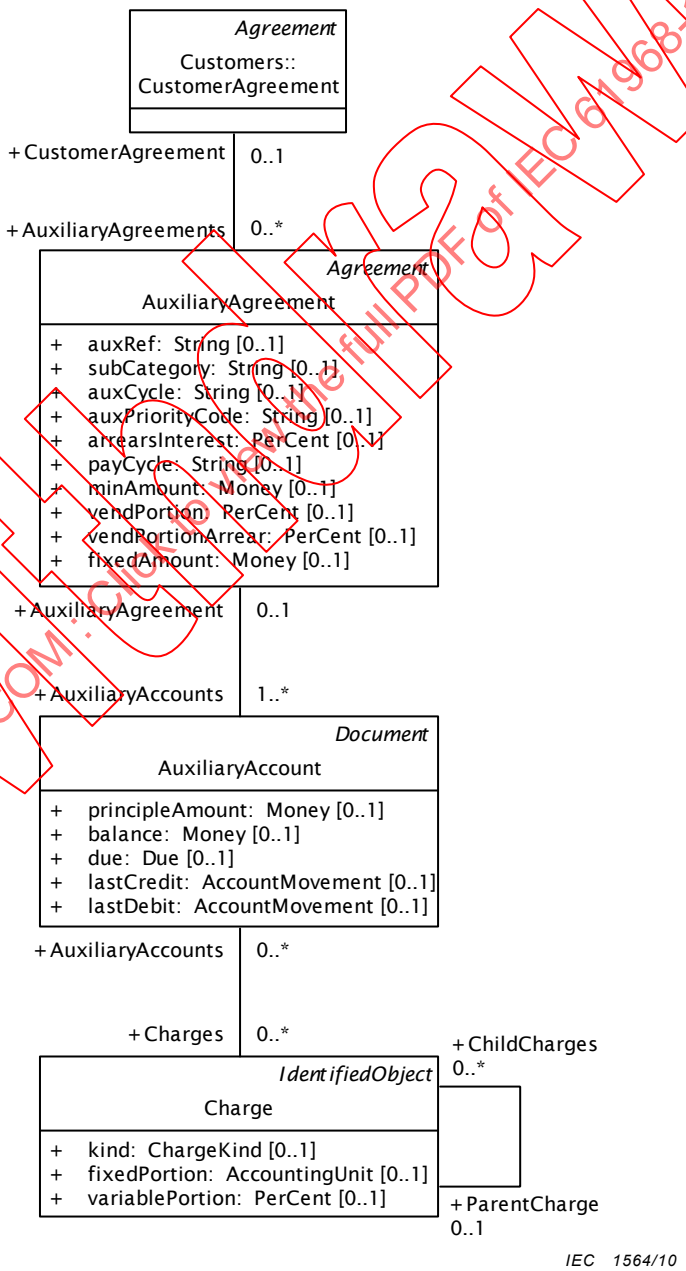


Figure 18 – DCIM auxiliary agreement model

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.

4.4.10.4 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 19.

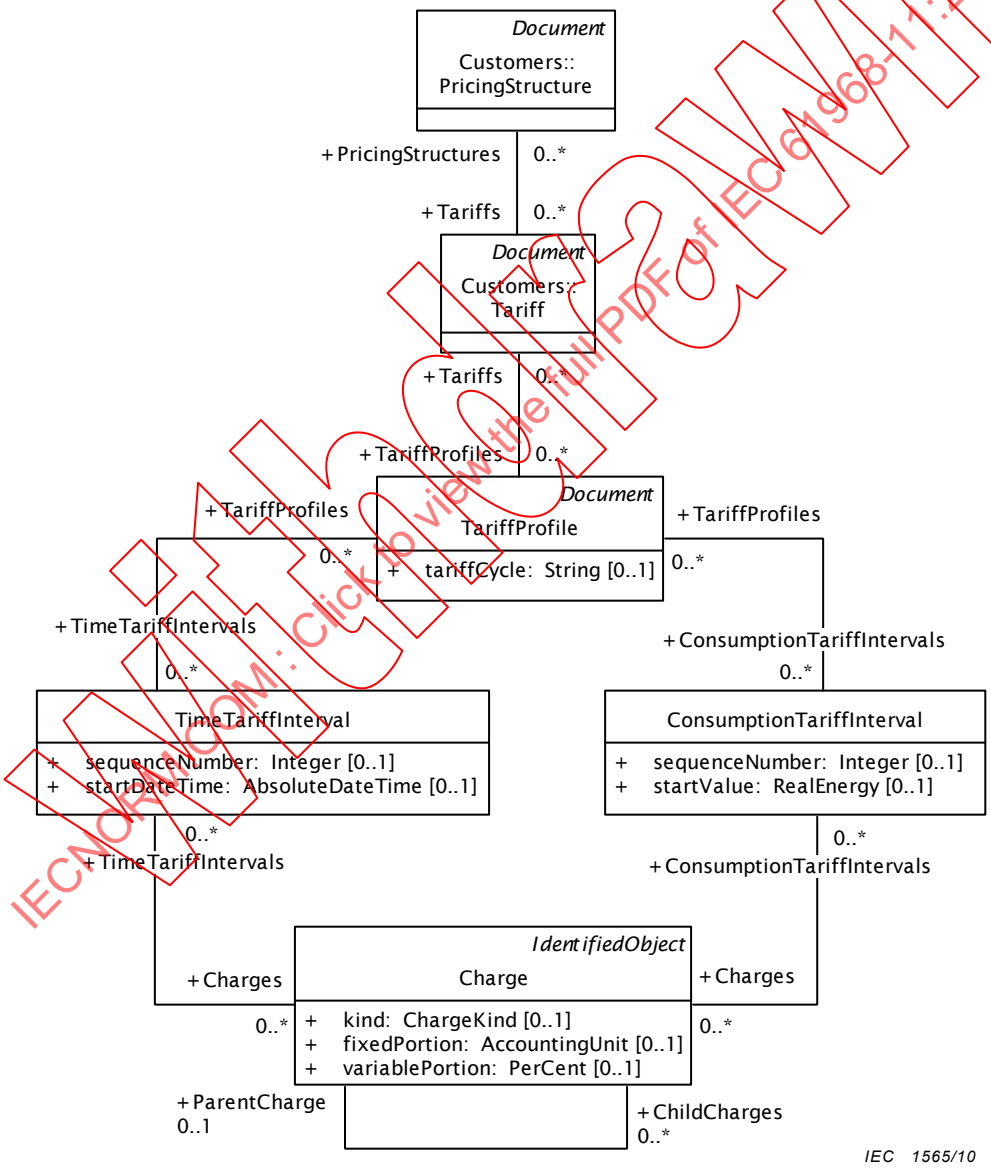


Figure 19 – DCIM pricing structure model

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 **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

Subclauses 4.5 to 4.8 of IEC 61970-301 describe CIM modelling tools, CIM extensions, and implementation conventions. The following points have changed since the last published edition 1.0 in 2003.

- Enterprise Architect is now used, instead of Rational Rose, to maintain the UML that defines the DCIM. The most current Enterprise Architect file (*.eap) provides the most current DCIM documentation.
- CIMTool provides a method of extending the DCIM and generating profiles, but other methods may be used as well.
- The **Naming** class is now the **IdentifiedObject** class. Interoperability testing has proven the need to maintain unique and persistent object identifiers across model domains. In practice, one way of achieving this would be the usage of universally unique identifiers (UUIDs) for the **mRID** attribute of **IdentifiedObject**. However, this is not the requirement.
- Older versions of CIM UML use three standard UML stereotypes: **Primitive**, enumeration and **Datatype**. The latter one is however used with a specific CIM semantics (not the standard UML semantics): for a triple of attributes {value, unit, multiplier}, which implies custom mapping to serialisation artefacts (RDFS, OWL, XSD). Therefore, a new custom UML stereotype, **Compound**, has been introduced to “fulfil” the semantics of standard UML dataType: group of values without identity. Classes with this stereotype never participate in relationships (generalisation, association), but are simply used as types for attributes.

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 Context

The CIM is partitioned into subpackages. Classes within the packages are listed alphabetically. Native class attributes are listed first, followed by inherited attributes. Native associations are listed first for each class, followed by inherited associations. The associations are described according to the role of each class participating in the 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 “note” column will contain text indicating the attributes is inherited from a specific class. The note 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 Attributes table, 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..*]	[0..*] OperatedBy_Companies	Company	inherited from: PowerSystemResource
[1]	[0..*] Contains_Measurements	Measurement	inherited from: PowerSystemResource
[1]	[0..*] OperatingShare	OperatingShare	inherited from: PowerSystemResource
[1..*]	[0..1] ModelingAuthoritySet	ModelingAuthoritySet	inherited from: IdentifiedObject

In the association ends table, the first column describes the multiplicity at this end of the association (i.e., how this class participates in association). 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 the type of each enum within the enumeration is not defined.

Table 4 – Enums documentation

literal	description
steel	
lead	
lock	
other	

6 Package architecture (normative)

6.1 General

This specification is automatically generated from the CIM model file iec61970cim14v13_iec61968cim10v31_combined.eap.

6.2 Top package IEC61968

The IEC 61968 subpackages of the CIM are developed, standardized and maintained by IEC TC57 Working Group 14: System interfaces for distribution management (WG14).

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

Figure 20 shows package diagram Main.

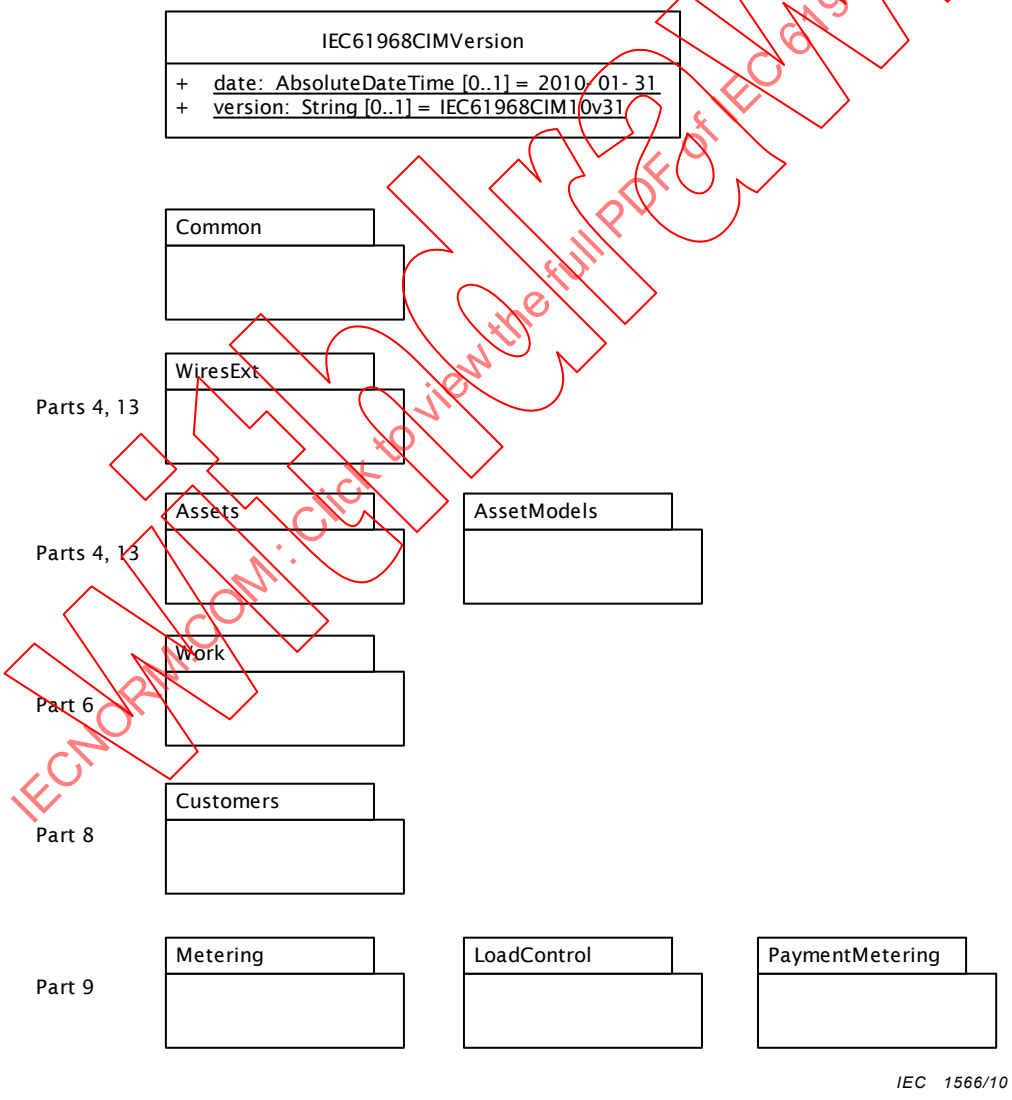


Figure 20 – Package diagram IEC61968::Main

This diagram shows version and normative contents of the CIM extensions for distribution.

Annotation on the left side of the diagram indicates the part of IEC 61968 that has been mainly driving modelling requirements for the respective package(s).

Figure 21 shows package diagram Dependencies.

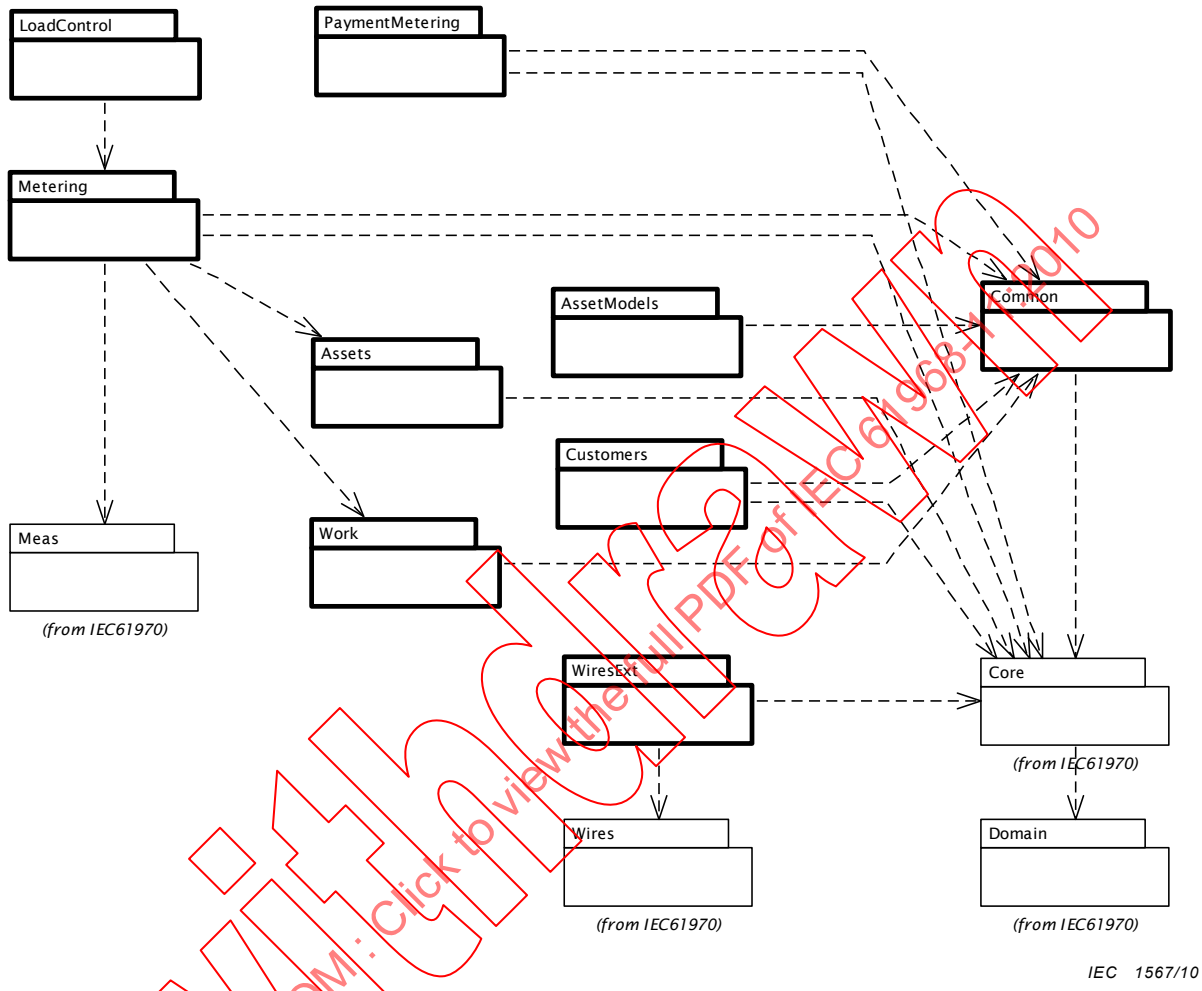


Figure 21 – Package diagram IEC61968::Dependencies

This diagram shows in bold contents of this package, as well as the dependencies based on inheritance only.

Figure 22 shows package diagram StdCIM.

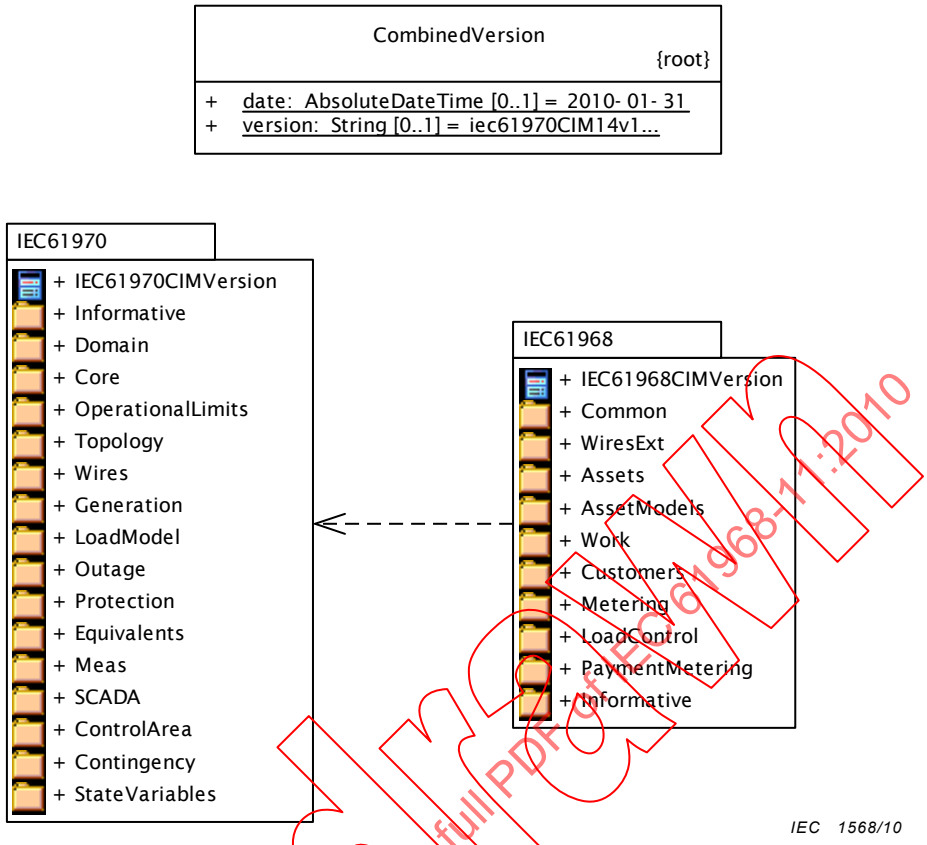


Figure 22 – Package diagram IEC61968::StdCIM

This diagram shows version and normative contents of the currently defined TC 57 CIM model.

Figure 23 shows logical diagram DCIMKeyClasses.

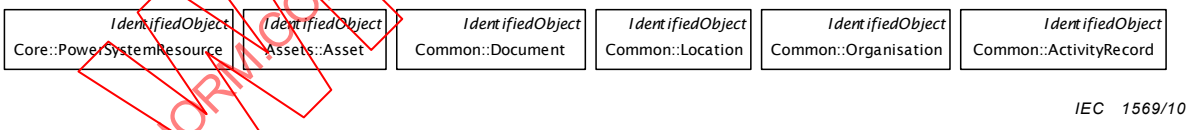


Figure 23 – Logical diagram IEC61968::DCIMKeyClasses

This diagram shows key classes in DCIM.

6.2.1 IEC61968CIMVersion

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=2010-01-31 (const)	AbsoluteDateTime	Form is YYYY-MM-DD for example for January 5, 2009 it is 2009-01-05.
version=IEC61968CIM10v31 (const)	String	Form is IEC61968CIMXXvYY where XX is the major CIM package version and the YY is the minor version. For example IEC61968CIM10v17.

6.2.2 Package Common

6.2.2.1 General

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

Figure 24 shows logical diagram CommonInheritance.

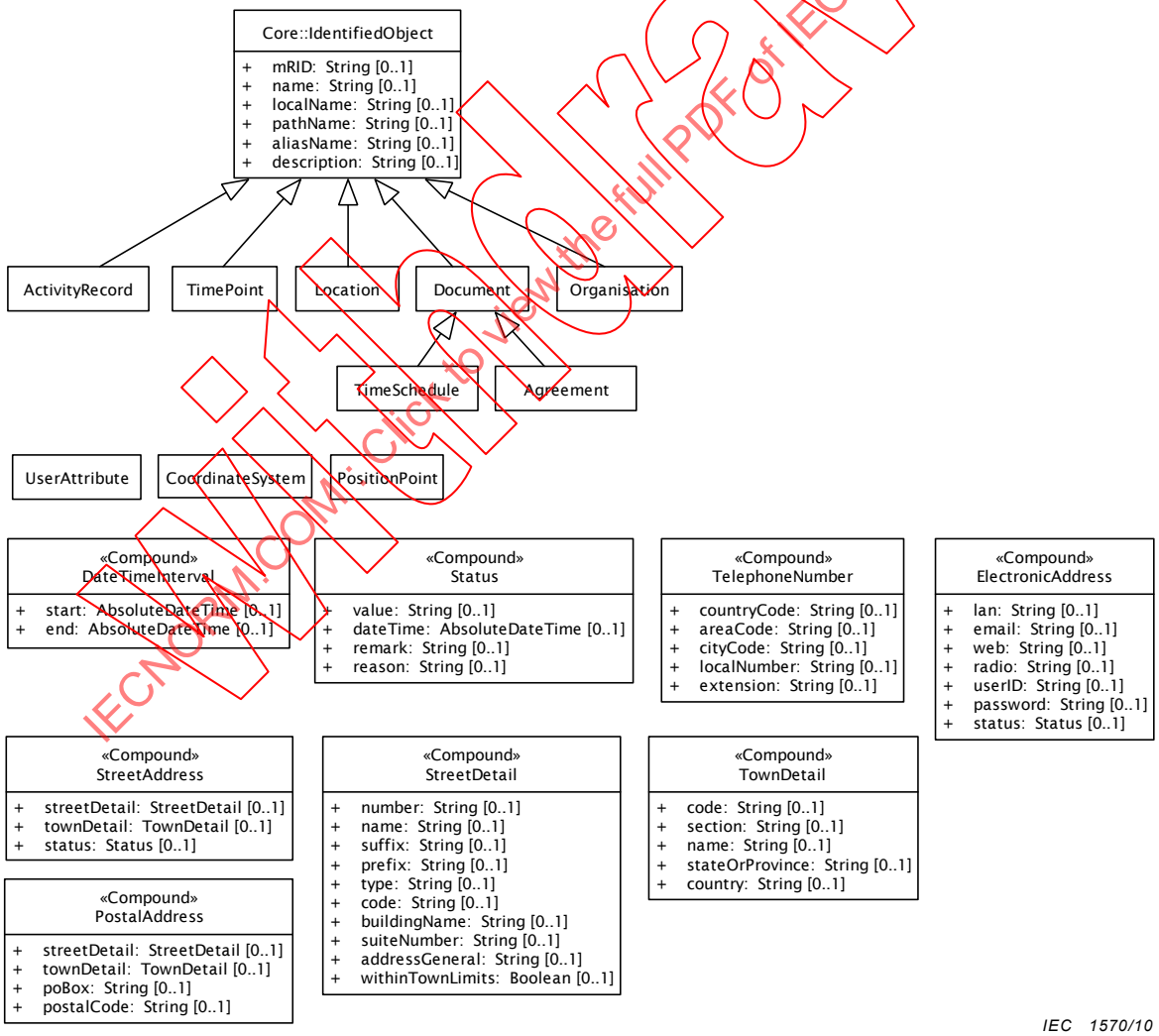


Figure 24 – Logical diagram Common::CommonInheritance

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

Figure 25 shows logical diagram CommonOverview.

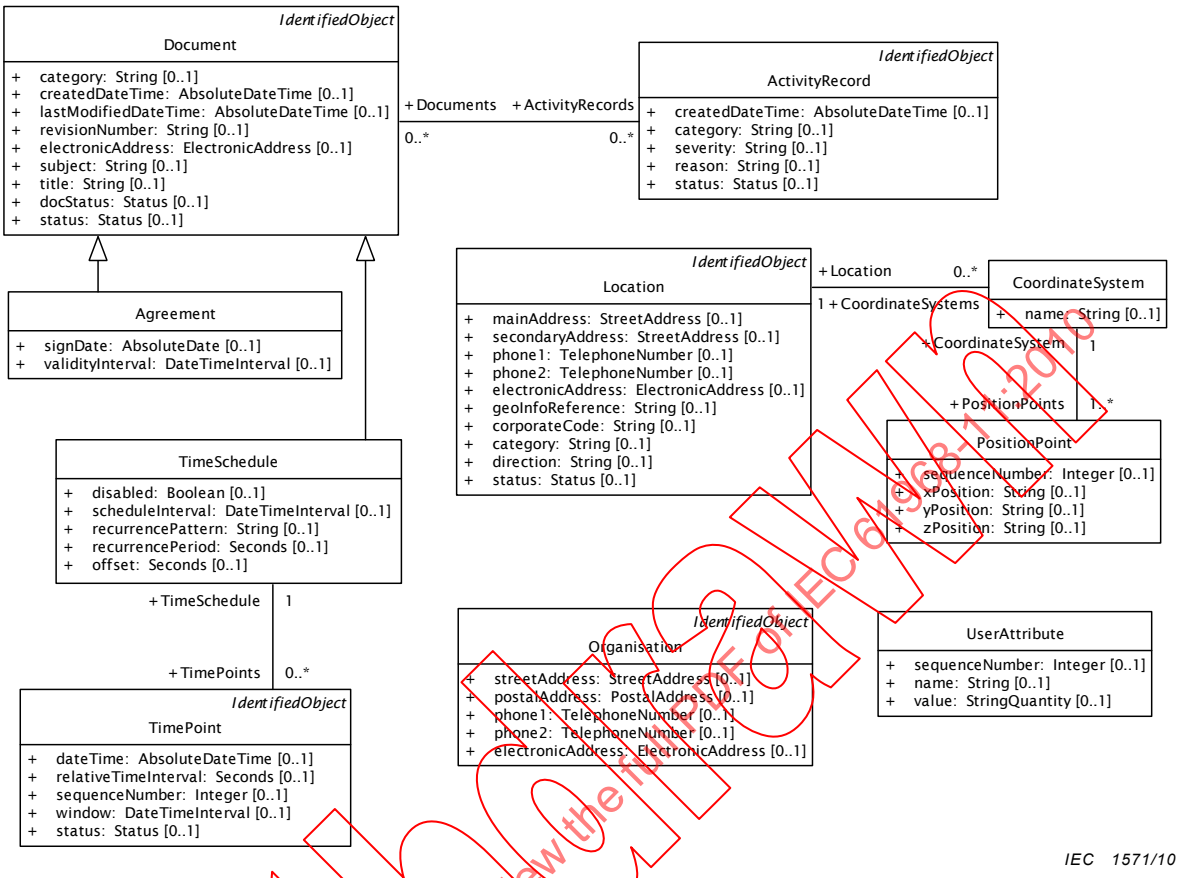


Figure 25 – Logical diagram Common::CommonOverview

This diagram shows normative classes from this package.

Figure 26 shows logical diagram DCIMLocations.

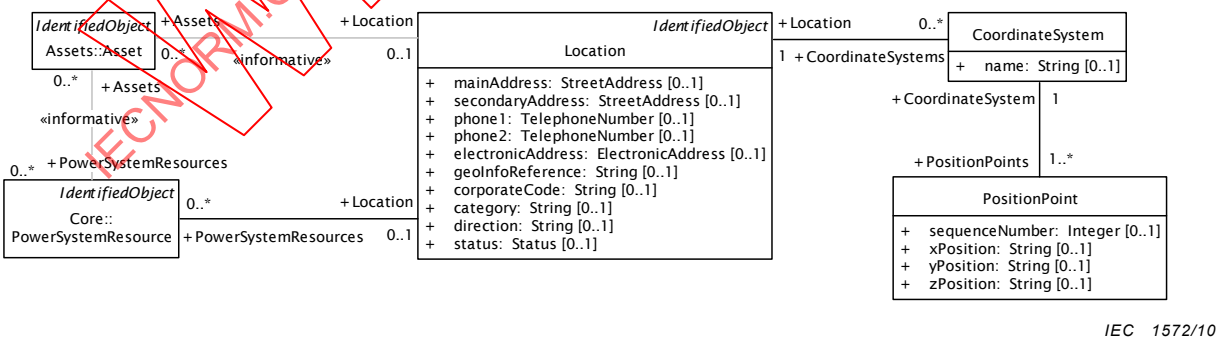


Figure 26 – Logical diagram Common::DCIMLocations

This diagram shows classes used to model locations of power system resources and assets in DCIM.

6.2.2.2 DateTimeInterval compound

Interval of date and time.

Table 6 shows all attributes of DateTimeInterval.

Table 6 – Attributes of Common::DateTimeInterval

name	type	description
start	AbsoluteDateTime	Date and time that this interval started.
end	AbsoluteDateTime	Date and time that this interval ended.

6.2.2.3 Status compound

Current status information relevant to an entity.

Table 7 shows all attributes of Status.

Table 7 – Attributes of Common::Status

name	type	description
value	String	Status value at 'dateTime'; prior status changes may have been kept in instances of ActivityRecords associated with the object to which this Status applies.
dateTime	AbsoluteDateTime	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'.

6.2.2.4 PostalAddress compound

General purpose postal address information.

Table 8 shows all attributes of PostalAddress.

Table 8 – 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.

6.2.2.5 StreetAddress compound

General purpose street address information.

Table 9 shows all attributes of StreetAddress.

Table 9 – Attributes of Common::StreetAddress

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

6.2.2.6 StreetDetail compound

Street details, in the context of address.

Table 10 shows all attributes of StreetDetail.

Table 10 – 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.
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).

6.2.2.7 TownDetail compound

Town details, in the context of address.

Table 11 shows all attributes of TownDetail.

Table 11 – 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.

6.2.2.8 ElectronicAddress compound

Electronic address information.

Table 12 shows all attributes of ElectronicAddress.

Table 12 – Attributes of Common::ElectronicAddress

name	type	description
lan	String	Address on local area network.
email	String	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.
status	Status	Status of this electronic address.

6.2.2.9 TelephoneNumber compound

Telephone number.

Table 13 shows all attributes of TelephoneNumber.

Table 13 – 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.

6.2.2.10 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 14 shows all attributes of ActivityRecord.

Table 14 – Attributes of Common::ActivityRecord

name	type	description
createdDateTime	AbsoluteDateTime	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).
category	String	Category of event resulting in this activity record.
severity	String	Severity level of event resulting in this activity

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

Table 15 shows all association ends of ActivityRecord with other classes.

Table 15 – 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..*]	[0..*] Documents	Document	All documents for which this activity record has been created.

6.2.2.11 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 16 shows all attributes of Agreement.

Table 16 – Attributes of Common::Agreement

name	type	description
signDate	AbsoluteDate	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).
category	String	inherited from: Document
createdDateTime	AbsoluteDateTime	inherited from: Document
lastModifiedDateTime	AbsoluteDateTime	inherited from: Document
revisionNumber	String	inherited from: Document
subject	String	inherited from: Document
title	String	inherited from: Document
docStatus	Status	inherited from: Document
status	Status	inherited from: Document
electronicAddress	ElectronicAddress	inherited from: Document
aliasName	String	inherited from: IdentifiedObject
description	String	inherited from: IdentifiedObject

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

Table 17 shows all association ends of Agreement with other classes.

Table 17 – Association ends of Common::Agreement with other classes

[mult from]	[mult to] name	type	description
[0..*]	[0..*] ActivityRecords	ActivityRecord	inherited from: Document

6.2.2.12 CoordinateSystem

Coordinate reference system.

Table 18 shows all attributes of CoordinateSystem.

Table 18 – Attributes of Common::CoordinateSystem

name	type	description
name	String	Name of this coordinate system.

Table 19 shows all association ends of CoordinateSystem with other classes.

Table 19 – Association ends of Common::CoordinateSystem with other classes

[mult from]	[mult to] name	type	description
[1..1]	[1..*] PositionPoints	PositionPoint	Sequence of position points expressed in this coordinate system.
[0..*]	[1..1] Location	Location	Location described by using position points in this coordinate system.

6.2.2.13 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 20 shows all attributes of Document.

Table 20 – Attributes of Common::Document

name	type	description
category	String	Utility-specific categorisation of this document, according to their corporate standards, practices, and existing IT systems (e.g., for

name	type	description
		management of assets, maintenance, work, outage, customers, etc.).
createdDateTime	AbsoluteDateTime	Date and time that this document was created.
lastModifiedDateTime	AbsoluteDateTime	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.
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.
electronicAddress	ElectronicAddress	Electronic address.
aliasName	String	inherited from: IdentifiedObject
description	String	inherited from: IdentifiedObject
localName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject
pathName	String	inherited from: IdentifiedObject

Table 21 shows all association ends of Document with other classes.

Table 21 – Association ends of Common::Document with other classes

[mult from]	[mult to] name	type	description
[0..*]	[0..*] ActivityRecords	ActivityRecord	All activity records created for this document.

6.2.2.14 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 is defined with one or more postition points (coordinates) in a given coordinate system.

Table 22 shows all attributes of Location.

Table 22 – Attributes of Common::Location

name	type	description
category	String	Category 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).
corporateCode	String	Utility-specific code for the location.

name	type	description
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'.
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 ServiceDeliveryPoint may be located on the second floor of an apartment building.
geoInfoReference	String	(if applicable) Reference to geographical information source, often external to the utility.
status	Status	Status of this location.
phone1	TelephoneNumber	Phone number.
phone2	TelephoneNumber	Additional phone number.
electronicAddress	ElectronicAddress	Electronic address.
aliasName	String	inherited from: IdentifiedObject
description	String	inherited from: IdentifiedObject
localName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject
pathName	String	inherited from: IdentifiedObject

Table 23 shows all association ends of Location with other classes.

Table 23 – 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.
[1..1]	[0..*] CoordinateSystems	CoordinateSystem	All coordinate systems used to describe position points of this location.

6.2.2.15 Organisation

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

Table 24 shows all attributes of Organisation.

Table 24 – 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.

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

6.2.2.16 PositionPoint

Set of spatial coordinates that determine a point. Use a single position point instance to desribe 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 25 shows all attributes of PositionPoint.

Table 25 – 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 26 shows all association ends of PositionPoint with other classes.

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

[mult from]	[mult to] name	type	description
[1..*]	[1..1] CoordinateSystem	CoordinateSystem	Coordinate system in which the coordinates of this position point are expressed.

6.2.2.17 TimePoint

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

Table 27 shows all attributes of TimePoint.

Table 27 – Attributes of Common::TimePoint

name	type	description
dateTime	AbsoluteDateTime	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

name	type	description
		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 to 5:00, etc.).
status	Status	Status of this time point.
aliasName	String	inherited from: IdentifiedObject
description	String	inherited from: IdentifiedObject
localName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject
pathName	String	inherited from: IdentifiedObject

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

Table 28 – 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.

6.2.2.18 TimeSchedule

Description of anything that changes through time. Time schedule is used to perform a single-valued function of time. Use inherited 'category' 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 29 shows all attributes of TimeSchedule.

Table 29 – 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).
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

name	type	description
		each hour.
category	String	inherited from: Document
createdDateTime	AbsoluteDateTime	inherited from: Document
lastModifiedDateTime	AbsoluteDateTime	inherited from: Document
revisionNumber	String	inherited from: Document
subject	String	inherited from: Document
title	String	inherited from: Document
docStatus	Status	inherited from: Document
status	Status	inherited from: Document
electronicAddress	ElectronicAddress	inherited from: Document
aliasName	String	inherited from: IdentifiedObject
description	String	inherited from: IdentifiedObject
localName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject
pathName	String	inherited from: IdentifiedObject

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

Table 30 – 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..*]	[0..*] ActivityRecords	ActivityRecord	inherited from: Document

6.2.2.19 UserAttribute

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 31 shows all attributes of UserAttribute.

Table 31 – 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 32 shows all association ends of UserAttribute with other classes.

Table 32 – 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.

6.2.3 Package WiresExt

6.2.3.1 General

This package contains the information classes that extend IEC61970::Wires package with power system resources required for distribution network modelling, including unbalanced networks.

Figure 27 shows logical diagram WiresExtInheritance.

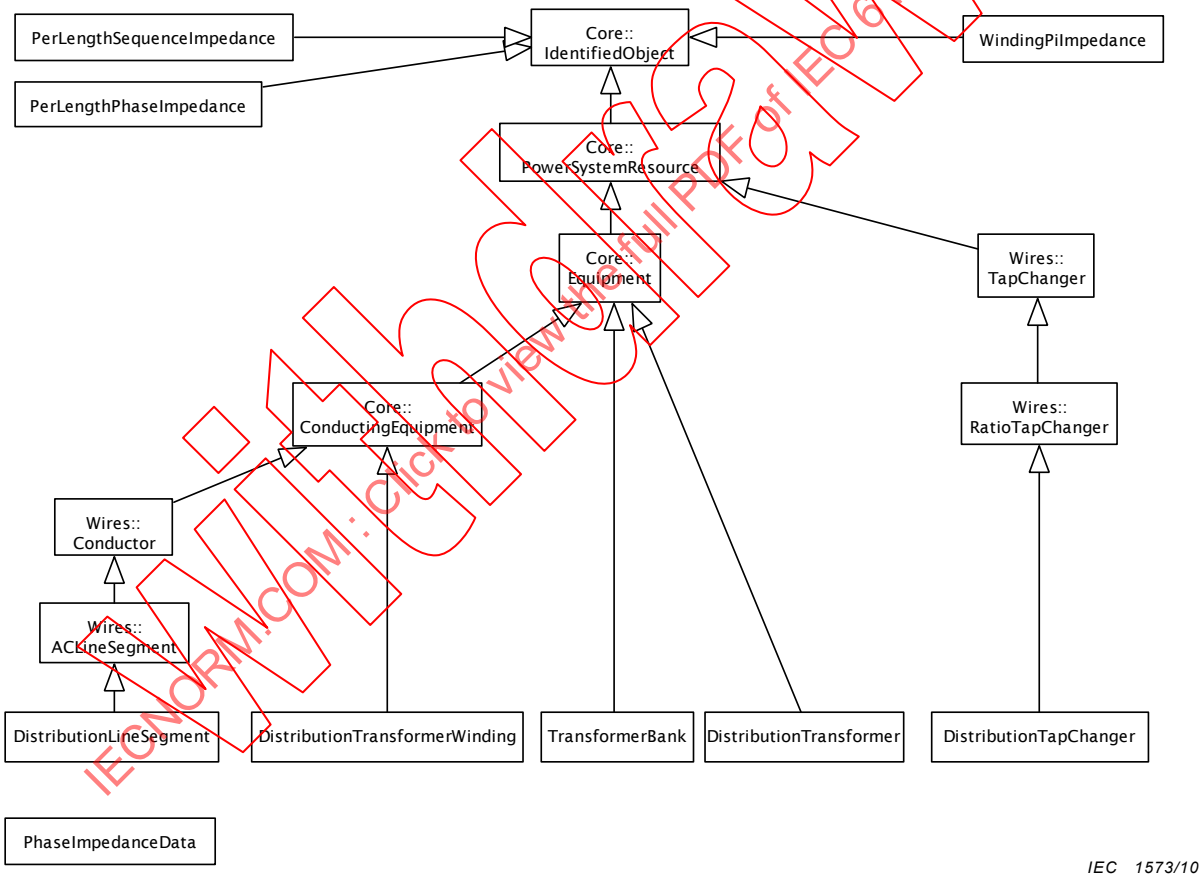


Figure 27 – Logical diagram WiresExt::WiresExtInheritance

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

Figure 28 shows logical diagram DCIMLoadModel.

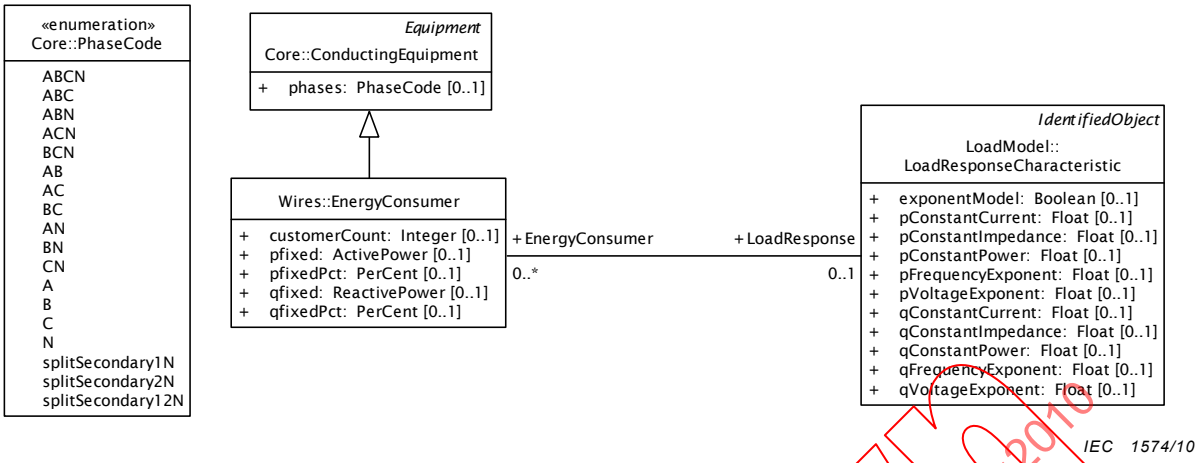


Figure 28 – Logical diagram WiresExt::DCIMLoadModel

This diagrams shows classes used to model loads in DCIM.

Figure 29 shows logical diagram DCIMLineModel.

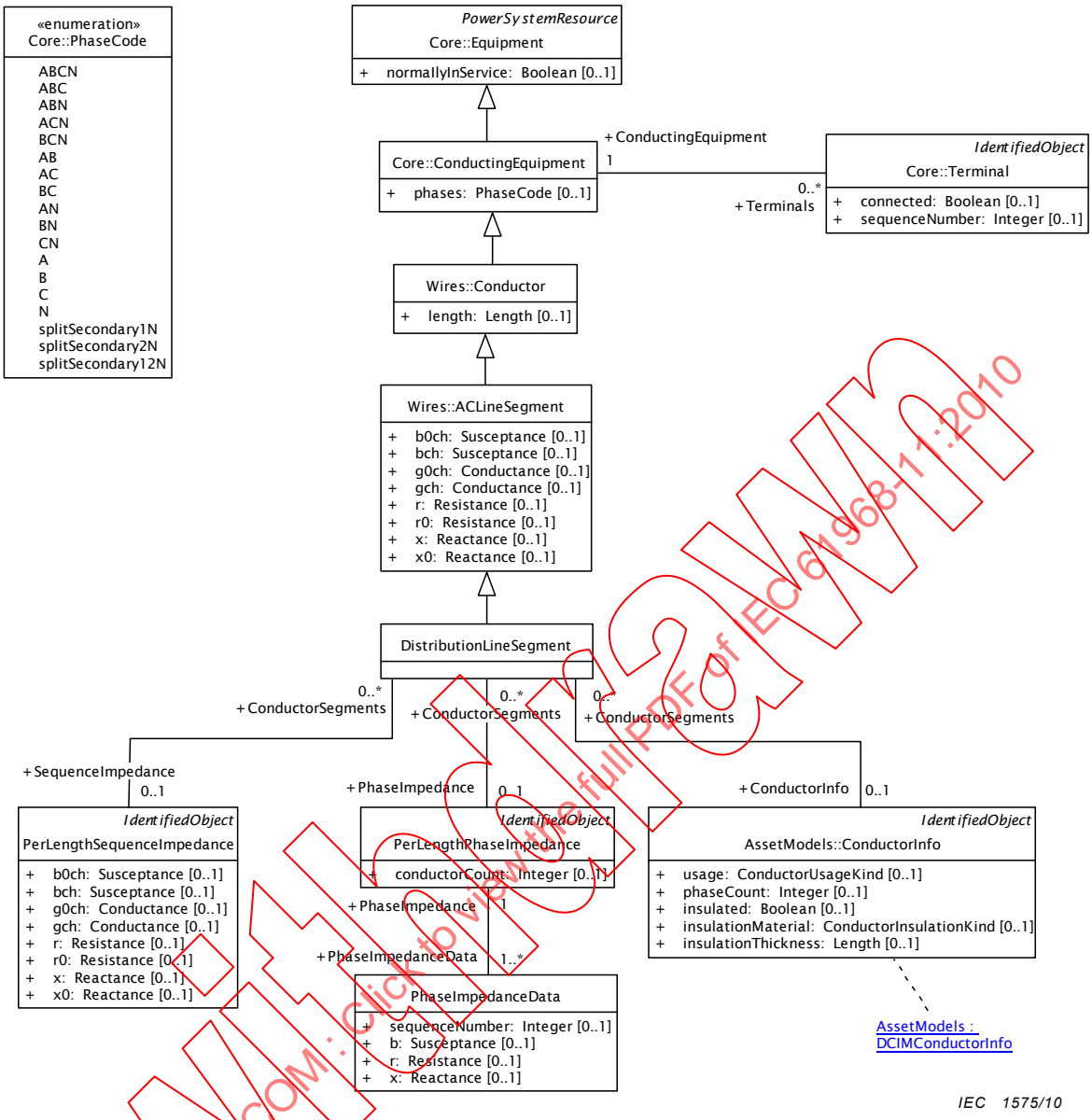


Figure 29 – Logical diagram WiresExt::DCIMLineModel

This diagram shows one part of classes used for modelling lines and circuits in DCIM.

Figure 30 shows logical diagram DCIMTransformerModel.

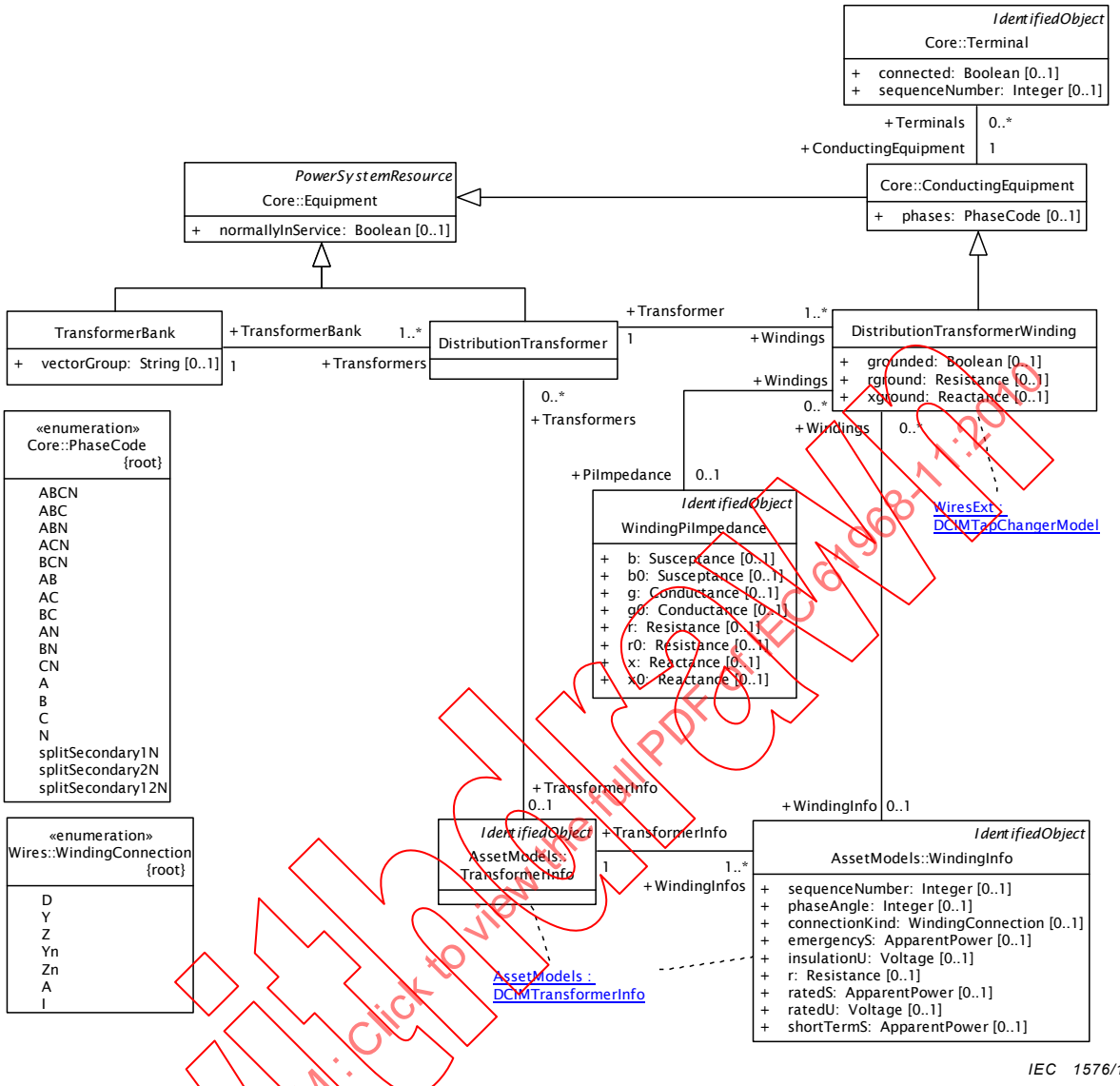


Figure 30 – Logical diagram WiresExt::DCIMTransformerModel

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

Figure 31 shows logical diagram DCIMTapChangerModel.

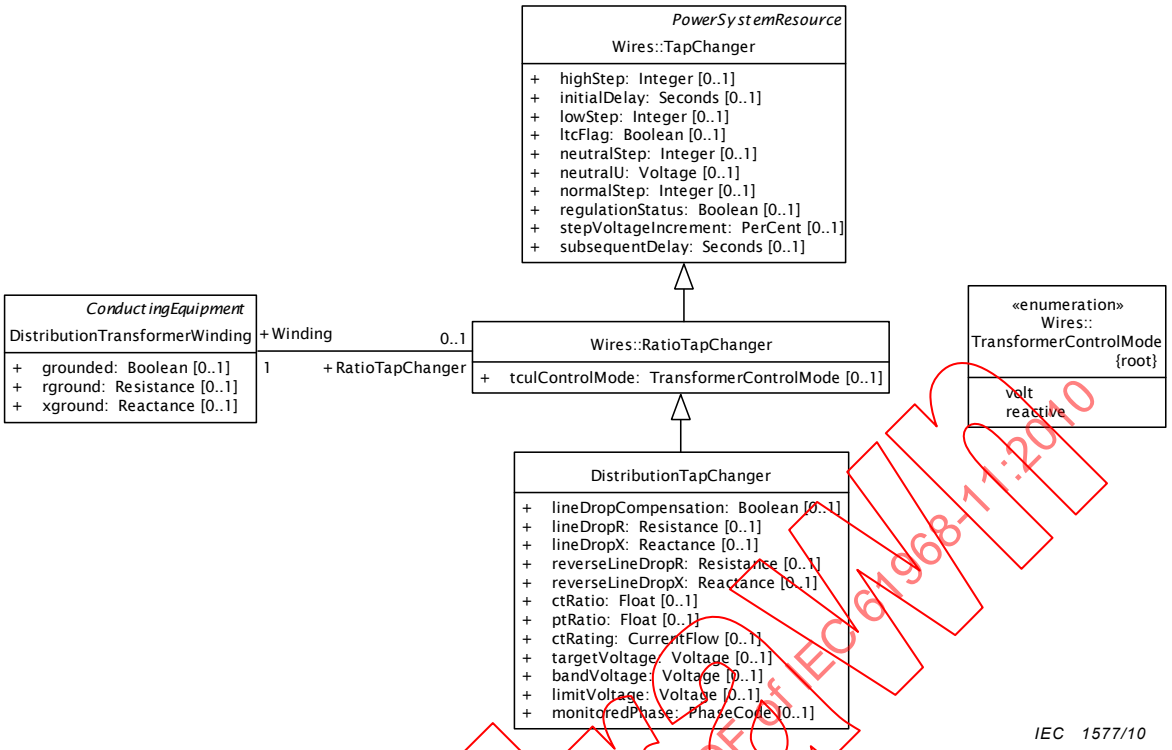


Figure 31 – Logical diagram WiresExt::DCIMTapChangerModel

This diagram shows classes used to model tap changers in DCIM.

6.2.3.2 DistributionLineSegment

Extends ACLineSegment with references to a library of standard types from which electrical parameters can be calculated, as follows:

– calculate electrical parameters from asset data, using associated ConductorInfo, with values then multiplied by Conductor.length to produce a matrix model;

– calculate unbalanced electrical parameters from associated PerLengthPhaseImpedance, then multiplied by Conductor.length to produce a matrix model;

– calculate transposed electrical parameters from associated PerLengthSequenceImpedance, then multiplied by Conductor.length to produce a sequence model.

For symmetrical, transposed 3ph lines, it is sufficient to use inherited ACLineSegment attributes, which describe sequence impedances and admittances for the entire length of the segment.

Known issue: Attributes expressing impedances and admittances in PerLengthSequenceImpedance and PhaseImpedanceData use Resistance, etc., which describe pre-calculated, full length of segment, while we should have a longitudinal unit, per length. Taking 'r' as example, its 'unit'=Ohm, but the value is effectively in Ohm/m, so the value needs to be multiplied by Conductor.length. This is against the whole idea of unit data types and is semantically wrong, but base CIM does not have the required data types at this moment. Until the revision of unit modelling in CIM, applications need to deduce and locally handle appending "/m" for units and ensure they multiply the values by Conductor.length.

Table 33 shows all attributes of DistributionLineSegment.

Table 33 – Attributes of WiresExt::DistributionLineSegment

name	type	description
b0ch	Susceptance	inherited from: ACLineSegment
bch	Susceptance	inherited from: ACLineSegment
g0ch	Conductance	inherited from: ACLineSegment
gch	Conductance	inherited from: ACLineSegment
r	Resistance	inherited from: ACLineSegment
r0	Resistance	inherited from: ACLineSegment
x	Reactance	inherited from: ACLineSegment
x0	Reactance	inherited from: ACLineSegment
length	Length	inherited from: Conductor
phases	PhaseCode	inherited from: ConductingEquipment
normallyInService	Boolean	inherited from: Equipment
aliasName	String	inherited from: IdentifiedObject
description	String	inherited from: IdentifiedObject
localName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject
pathName	String	inherited from: IdentifiedObject

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

Table 34 – Association ends of WiresExt::DistributionLineSegment with other classes

[mult from]	[mult to] name	type	description
[0..*]	[0..1] PhaseImpedance	PerLengthPhaseImpedance	Phase impedance of this conductor segment; used for unbalanced model.
[0..*]	[0..1] SequenceImpedance	PerLengthSequenceImpedance	Sequence impedance of this conductor segment; used for balanced model.
[0..*]	[0..1] ConductorInfo	ConductorInfo	Conductor data of this conductor segment.
[0..*]	[0..1] BaseVoltage	BaseVoltage	inherited from: ConductingEquipment
[1..1]	[0..*] Terminals	Terminal	inherited from: ConductingEquipment
[0..*]	[0..*] ProtectionEquipments	ProtectionEquipment	inherited from: ConductingEquipment
[1..1]	[0..*] ClearanceTags	ClearanceTag	inherited from: ConductingEquipment
[1..1]	[0..1] SvStatus	SvStatus	inherited from: ConductingEquipment
[0..*]	[0..1] EquipmentContainer	EquipmentContainer	inherited from: Equipment
[1..1]	[0..*] OperationalLimitSet	OperationalLimitSet	inherited from: Equipment
[1..1]	[0..*] ContingencyEquipment	ContingencyEquipment	inherited from: Equipment
[0..*]	[0..1] PSRType	PSRType	inherited from: PowerSystemResource
[0..1]	[0..*] Measurements	Measurement	inherited from: PowerSystemResource
[1..1]	[0..*] OperatingShare	OperatingShare	inherited from: PowerSystemResource
[0..*]	[0..*] PsrLists	PsrList	inherited from: PowerSystemResource
[1..1]	[0..1] OutageSchedule	OutageSchedule	inherited from: PowerSystemResource
[0..*]	[0..*] ReportingGroup	ReportingGroup	inherited from: PowerSystemResource
[0..*]	[0..1] Location	Location	inherited from: PowerSystemResource

6.2.3.3 DistributionTapChanger

Additional ratio tap changer parameters common to distribution line regulators. 'tculControlMode' would always be 'volt'.

If 'monitoredPhase' is not specified, then if the controlled DistributionTransformerWinding is single-phase, the PT primary is assumed to be connected across that winding, which is the normal case. If the controlled winding is three-phase, then the 'monitoredPhase' is assumed to be 'AN', unless otherwise specified.

Whenever 'ctRatio' and 'ptRatio' are specified, it's customary to specify the R and X in "volts" referred to the PT secondary circuit, otherwise R and X are in feeder primary ohms.

If 'ptRatio' is not specified, then 'targetVoltage', 'limitVoltage', and 'bandVoltage' are on the feeder primary base, phase-neutral or phase-phase depending on the 'monitoredPhase'. Otherwise, these attributes are all on the PT secondary base.

Table 35 shows all attributes of DistributionTapChanger.

Table 35 – Attributes of WiresExt::DistributionTapChanger

name	type	description
ctRatio	Float	Built-in current transducer ratio.
ptRatio	Float	Built-in voltage transducer ratio.
reverseLineDropR	Resistance	Line drop compensator resistance setting for reverse power flow.
reverseLineDropX	Reactance	Line drop compensator reactance setting for reverse power flow.
lineDropCompensation	Boolean	If true, the line drop compensation is to be applied.
lineDropR	Resistance	Line drop compensator resistance setting for normal (forward) power flow.
lineDropX	Reactance	Line drop compensator reactance setting for normal (forward) power flow.
targetVoltage	Voltage	Target voltage on the PT secondary base.
bandVoltage	Voltage	Voltage range (max - min) on the PT secondary base, centered on 'targetVoltage'.
limitVoltage	Voltage	Maximum allowed regulated voltage on the PT secondary base, regardless of line drop compensation. Sometimes referred to as first-house protection.
monitoredPhase	PhaseCode	Phase voltage controlling this regulator, measured at regulator location.
ctRating	CurrentFlow	Built-in current transformer primary rating.
tculControlMode	TransformerControlMode	inherited from: RatioTapChanger
highStep	Integer	inherited from: TapChanger
initialDelay	Seconds	inherited from: TapChanger
lowStep	Integer	inherited from: TapChanger
neutralStep	Integer	inherited from: TapChanger
neutralU	Voltage	inherited from: TapChanger
normalStep	Integer	inherited from: TapChanger
stepVoltageIncrement	PerCent	inherited from: TapChanger
subsequentDelay	Seconds	inherited from: TapChanger
ltcFlag	Boolean	inherited from: TapChanger

name	type	description
regulationStatus	Boolean	inherited from: TapChanger
aliasName	String	inherited from: IdentifiedObject
description	String	inherited from: IdentifiedObject
localName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject
pathName	String	inherited from: IdentifiedObject

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

Table 36 – Association ends of WiresExt::DistributionTapChanger with other classes

[mult from]	[mult to] name	type	description
[0..1]	[1..1] TransformerWinding	TransformerWinding	inherited from: RatioTapChanger
[1..1]	[0..1] RatioVariationCurve	RatioVariationCurve	inherited from: RatioTapChanger
[0..1]	[1..1] Winding	DistributionTransformerWinding	inherited from: RatioTapChanger
[0..*]	[0..1] RegulatingControl	RegulatingControl	inherited from: TapChanger
[1..1]	[0..1] SvTapStep	SvTapStep	inherited from: TapChanger
[1..1]	[0..1] ImpedanceVariationCurve	ImpedanceVariationCurve	inherited from: TapChanger
[1..1]	[0..*] TapSchedules	TapSchedule	inherited from: TapChanger
[0..*]	[0..1] PSRType	PSRType	inherited from: PowerSystemResource
[0..1]	[0..*] Measurements	Measurement	inherited from: PowerSystemResource
[1..1]	[0..*] OperatingShare	OperatingShare	inherited from: PowerSystemResource
[0..*]	[0..*] PsrLists	PsrList	inherited from: PowerSystemResource
[1..1]	[0..1] OutageSchedule	OutageSchedule	inherited from: PowerSystemResource
[0..*]	[0..*] ReportingGroup	ReportingGroup	inherited from: PowerSystemResource
[0..*]	[0..1] Location	Location	inherited from: PowerSystemResource

6.2.3.4 DistributionTransformer

An assembly of two or more coupled windings that transform electrical power between voltage levels. Supports both balanced and unbalanced winding connections.

This class differs from Wires::PowerTransformer as follows:

– it is part of a TransformerBank;
– it draws parameters exclusively from TransformerInfo and its associated classes.

Table 37 shows all attributes of DistributionTransformer.

Table 37 – Attributes of WiresExt::DistributionTransformer

name	type	description
normallyInService	Boolean	inherited from: Equipment
aliasName	String	inherited from: IdentifiedObject

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

Table 38 shows all association ends of DistributionTransformer with other classes.

Table 38 – Association ends of WiresExt::DistributionTransformer with other classes

[mult from]	[mult to] name	type	description
[1..*]	[1..1] TransformerBank	TransformerBank	Bank this transformer belongs to.
[1..1]	[1..*] Windings	DistributionTransformerWinding	All windings of this transformer.
[0..*]	[0..1] TransformerInfo	TransformerInfo	Transformer data.
[0..*]	[0..1] EquipmentContainer	EquipmentContainer	inherited from: Equipment
[1..1]	[0..*] OperationalLimitSet	OperationalLimitSet	inherited from: Equipment
[1..1]	[0..*] ContingencyEquipment	ContingencyEquipment	inherited from: Equipment
[0..*]	[0..1] PSRType	PSRType	inherited from: PowerSystemResource
[0..1]	[0..*] Measurements	Measurement	inherited from: PowerSystemResource
[1..1]	[0..*] OperatingShare	OperatingShare	inherited from: PowerSystemResource
[0..*]	[0..*] PsrLists	PsrList	inherited from: PowerSystemResource
[1..1]	[0..1] OutageSchedule	OutageSchedule	inherited from: PowerSystemResource
[0..*]	[0..*] ReportingGroup	ReportingGroup	inherited from: PowerSystemResource
[0..*]	[0..1] Location	Location	inherited from: PowerSystemResource

6.2.3.5 DistributionTransformerWinding

Conducting connection point of a distribution / unbalanced transformer winding instance.

This class differs from Wires::TransformerWinding as follows:

– the eight Pi model attributes are moved into separate class, that can be optionally referred to from several winding instances;

– the three grounding attributes can differ per winding instance, even for windings that use the same TransformerInfo, so they are kept on DistributionTransformerWinding;

– 'windingType' attribute is replaced by 'sequenceNumber' attribute on WindingInfo class;

– all the other attributes come from the WindingInfo (and its relationships). TransformerInfo is associated to the DistributionTransformer as referenceable data, so it can be defined once and referred to from instances, instead of being specified with each instance.

Table 39 shows all attributes of DistributionTransformerWinding.

Table 39 – Attributes of WiresExt::DistributionTransformerWinding

name	type	description
grounded	Boolean	(for Yn and Zn connections) True if the neutral is solidly grounded.
rground	Resistance	(for Yn and Zn connections) Resistance part of neutral impedance where 'grounded' is true.
xground	Reactance	(for Yn and Zn connections) Reactive part of neutral impedance where 'grounded' is true.
phases	PhaseCode	inherited from: ConductingEquipment
normallyInService	Boolean	inherited from: Equipment
aliasName	String	inherited from: IdentifiedObject
description	String	inherited from: IdentifiedObject
localName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject
pathName	String	inherited from: IdentifiedObject

Table 40 shows all association ends of DistributionTransformerWinding with other classes.

Table 40 – Association ends of WiresExt::DistributionTransformerWinding with other classes

[mult from]	[mult to] name	type	description
[0..*]	[0..1] PiImpedance	WindingPiImpedance	(accurate for 2- or 3-winding transformers only) Pi-model impedances of this winding.
[1..*]	[1..1] Transformer	DistributionTransformer	Transformer this winding belongs to.
[1..1]	[0..1] RatioTapChanger	RatioTapChanger	Ratio tap changer associated with this winding.
[1..1]	[0..1] PhaseTapChanger	PhaseTapChanger	Phase tap changer associated with this winding.
[0..*]	[0..1] WindingInfo	WindingInfo	Data for this winding.
[0..*]	[0..1] BaseVoltage	BaseVoltage	inherited from: ConductingEquipment
[1..1]	[0..*] Terminals	Terminal	inherited from: ConductingEquipment
[0..*]	[0..*] ProtectionEquipments	ProtectionEquipment	inherited from: ConductingEquipment
[1..1]	[0..*] ClearanceTags	ClearanceTag	inherited from: ConductingEquipment
[1..1]	[0..1] SvStatus	SvStatus	inherited from: ConductingEquipment
[0..*]	[0..1] EquipmentContainer	EquipmentContainer	inherited from: Equipment
[1..1]	[0..*] OperationalLimitSet	OperationalLimitSet	inherited from: Equipment
[1..1]	[0..*] ContingencyEquipment	ContingencyEquipment	inherited from: Equipment
[0..*]	[0..1] PSRType	PSRType	inherited from: PowerSystemResource
[0..1]	[0..*] Measurements	Measurement	inherited from: PowerSystemResource
[1..1]	[0..*] OperatingShare	OperatingShare	inherited from: PowerSystemResource
[0..*]	[0..*] PsrLists	PsrList	inherited from: PowerSystemResource
[1..1]	[0..1] OutageSchedule	OutageSchedule	inherited from: PowerSystemResource
[0..*]	[0..*] ReportingGroup	ReportingGroup	inherited from: PowerSystemResource
[0..*]	[0..1] Location	Location	inherited from: PowerSystemResource

6.2.3.6 PerLengthPhaseImpedance

Impedance and admittance parameters per unit length for n-wire unbalanced lines, in matrix form.

Table 41 shows all attributes of PerLengthPhaseImpedance.

Table 41 – Attributes of WiresExt::PerLengthPhaseImpedance

name	type	description
conductorCount	Integer	Number of phase, neutral, and other wires retained. Constrains the number of matrix elements and the phase codes that can be used with this matrix.
aliasName	String	inherited from: IdentifiedObject
description	String	inherited from: IdentifiedObject
localName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject
pathName	String	inherited from: IdentifiedObject

Table 42 shows all association ends of PerLengthPhaseImpedance with other classes.

Table 42 – Association ends of WiresExt::PerLengthPhaseImpedance with other classes

[mult from]	[mult to] name	type	description
[1..1]	[1..*] PhaseImpedanceData	PhaseImpedanceData	All data that belong to this conductor phase impedance.
[0..1]	[0..*] ConductorSegments	DistributionLineSegment	All conductor segments described by this phase impedance.

6.2.3.7 PerLengthSequenceImpedance

Sequence impedance and admittance parameters per unit length, for transposed lines of 1, 2, or 3 phases. For 1-phase lines, define x = x0 = xself. For 2-phase lines, define x = xs – xm and x0 = xs + xm.

Table 43 shows all attributes of PerLengthSequenceImpedance.

Table 43 – Attributes of WiresExt::PerLengthSequenceImpedance

name	type	description
b0ch	Susceptance	Zero sequence shunt (charging) susceptance, per unit of length.
bch	Susceptance	Positive sequence shunt (charging) susceptance, per unit of length.
g0ch	Conductance	Zero sequence shunt (charging) conductance, per unit of length.
gch	Conductance	Positive sequence shunt (charging) conductance, per unit of length.
r	Resistance	Positive sequence series resistance, per unit of length.

name	type	description
r0	Resistance	Zero sequence series resistance, per unit of length.
x	Reactance	Positive sequence series reactance, per unit of length.
x0	Reactance	Zero sequence series reactance, per unit of length.
aliasName	String	inherited from: IdentifiedObject
description	String	inherited from: IdentifiedObject
localName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject
pathName	String	inherited from: IdentifiedObject

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

Table 44 – Association ends of WiresExt::PerLengthSequenceImpedance with other classes

[mult from]	[mult to] name	type	description
[0..1]	[0..*] ConductorSegments	DistributionLineSegment	All conductor segments described by this sequence impedance.

6.2.3.8 PhaseImpedanceData

Triplet of resistance, reactance, and susceptance matrix element values.

Table 45 shows all attributes of PhaseImpedanceData.

Table 45 – Attributes of WiresExt::PhaseImpedanceData

name	type	description
sequenceNumber	Integer	Column-wise element index, assuming a symmetrical matrix. Ranges from 1 to N + N*(N-1)/2.
b	Susceptance	Susceptance matrix element value, per length of unit.
r	Resistance	Resistance matrix element value, per length of unit.
x	Reactance	Reactance matrix element value, per length of unit.

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

Table 46 – Association ends of WiresExt::PhaseImpedanceData with other classes

[mult from]	[mult to] name	type	description
[1..*]	[1..1] PhaseImpedance	PerLengthPhaseImpedance	Conductor phase impedance to which this data belongs.

6.2.3.9 TransformerBank

An assembly of transformers that are connected together. For three-phase transformers, there would be one transformer per bank. For banks of single-phase transformers, there will be more than one transformer per bank, and they need not be identical.

Table 47 shows all attributes of TransformerBank.

Table 47 – Attributes of WiresExt::TransformerBank

name	type	description
vectorGroup	String	Vector group of the bank for protective relaying, e.g., Dyn1. For unbalanced transformers, this may not be simply determined from the constituent winding connections.
normallyInService	Boolean	inherited from: Equipment
aliasName	String	inherited from: IdentifiedObject
description	String	inherited from: IdentifiedObject
localName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject
pathName	String	inherited from: IdentifiedObject

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

Table 48 – Association ends of WiresExt::TransformerBank with other classes

[mult from]	[mult to] name	type	description
[1..1]	[1..*] Transformers	DistributionTransformer	All transformers that belong to this bank.
[0..*]	[0..1] EquipmentContainer	EquipmentContainer	inherited from: Equipment
[1..1]	[0..*] OperationalLimitSet	OperationalLimitSet	inherited from: Equipment
[1..1]	[0..*] ContingencyEquipment	ContingencyEquipment	inherited from: Equipment
[0..*]	[0..1] PSRType	PSRType	inherited from: PowerSystemResource
[0..1]	[0..*] Measurements	Measurement	inherited from: PowerSystemResource
[1..1]	[0..*] OperatingShare	OperatingShare	inherited from: PowerSystemResource
[0..*]	[0..*] PsrLists	PsrList	inherited from: PowerSystemResource
[1..1]	[0..1] OutageSchedule	OutageSchedule	inherited from: PowerSystemResource
[0..*]	[0..*] ReportingGroup	ReportingGroup	inherited from: PowerSystemResource
[0..*]	[0..1] Location	Location	inherited from: PowerSystemResource

6.2.3.10 WindingPiImpedance

Transformer Pi-model impedance that accurately reflects impedance for transformers with 2 or 3 windings. For transformers with 4 or more windings, you must use TransformerInfo.

Table 49 shows all attributes of WindingPiImpedance.

Table 49 – Attributes of WiresExt::WindingPiImpedance

name	type	description
b	Susceptance	Magnetizing branch susceptance (B mag). The value can be positive or negative.
b0	Susceptance	Zero sequence magnetizing branch susceptance.
g	Conductance	Magnetizing branch conductance (G mag).
g0	Conductance	Zero sequence magnetizing branch conductance.
r	Resistance	DC resistance of the winding.

name	type	description
r0	Resistance	Zero sequence series resistance of the winding.
x	Reactance	Positive sequence series reactance of the winding. For a two winding transformer, the full reactance of the transformer should be entered on the primary (high voltage) winding.
x0	Reactance	Zero sequence series reactance of the winding.
aliasName	String	inherited from: IdentifiedObject
description	String	inherited from: IdentifiedObject
localName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject
pathName	String	inherited from: IdentifiedObject

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

Table 50 – Association ends of WiresExt::WindingPiImpedance with other classes

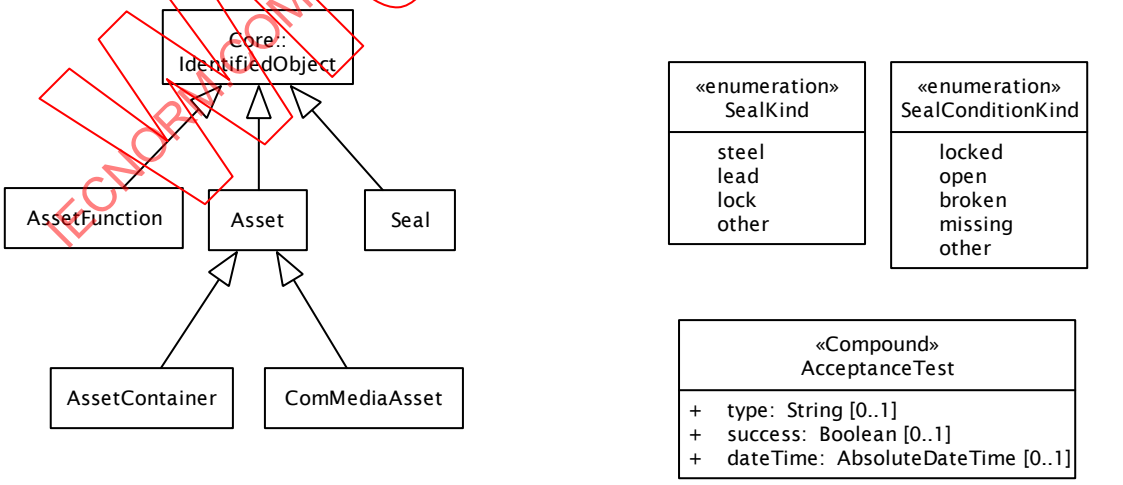
[mult from]	[mult to] name	type	description
[0..1]	[0..*] Windings	DistributionTransformerWinding	All windings having this Pi impedance.

6.2.4 Package Assets

6.2.4.1 General

This package contains the core information classes that support asset management applications with specialized classes for asset-level models for objects (as opposed to power system resource models, mainly defined in IEC61970::Wires package).

Figure 32 shows logical diagram AssetsInheritance.



IEC 1578/10

Figure 32 – Logical diagram Assets::AssetsInheritance

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

Figure 33 shows logical diagram AssetsOverview.

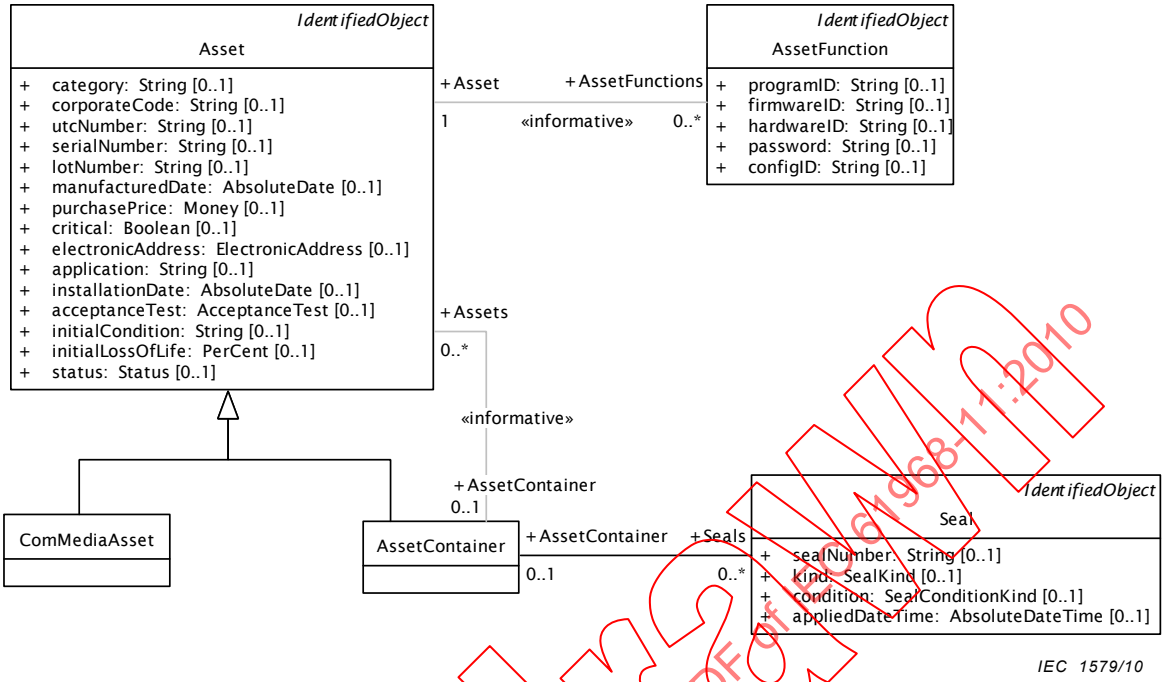


Figure 33 – Logical diagram Assets::AssetsOverview

This diagram shows normative classes from this package.

Figure 34 shows logical diagram DCIMAssetsAndPSRs.

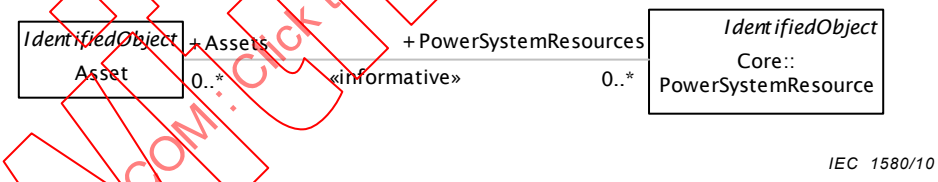


Figure 34 – Logical diagram Assets::DCIMAssetsAndPSRs

This diagram shows relationships between power system resource view and asset view of equipment in DCIM.

6.2.4.2 AcceptanceTest compound

Acceptance test for assets.

Table 51 shows all attributes of AcceptanceTest.

Table 51 – 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 life cycle, the date of each acceptance

name	type	description
		test may be recorded in Asset.ActivityRecord.status.dateTime.
dateTime	AbsoluteDateTime	Date and time the asset was last tested using the 'type' of test and yielding the current status in 'success' attribute.

6.2.4.3 SealConditionKind enumeration

Kind of seal condition.

Table 52 shows all literals of SealConditionKind.

Table 52 – Literals of Assets::SealConditionKind

literal	description
locked	
open	
broken	
missing	
other	

6.2.4.4 SealKind enumeration

Kind of seal.

Table 53 shows all literals of SealKind.

Table 53 – Literals of Assets::SealKind

literal	description
steel	
lead	
lock	
other	

6.2.4.5 Asset

Tangible resource of the utility, including power system equipment, 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 54 shows all attributes of Asset.

Table 54 – Attributes of Assets::Asset

name	type	description
category	String	Extension mechanism to accommodate utility-specific categorisation of Asset and its subtypes, according to their corporate standards, practices, and existing IT systems (e.g., for

name	type	description
		management of assets, maintenance, work, outage, customers, etc.).
corporateCode	String	Code for this type of asset.
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.
manufacturedDate	AbsoluteDate	Date this asset was manufactured.
purchasePrice	Money	Purchase price of asset.
critical	Boolean	True if asset is considered critical for some reason (for example, a pole with critical attachments).
application	String	The way this particular asset is being used in this installation. For example, the application of a bushing when attached to a specific transformer winding would be one of the following: H1, H2, H3, H0, X1, X2, X3, X0, Y1, Y2, Y3, Y0.
installationDate	AbsoluteDate	(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).
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.
electronicAddress	ElectronicAddress	Electronic address.
aliasName	String	inherited from: IdentifiedObject
description	String	inherited from: IdentifiedObject
localName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject
pathName	String	inherited from: IdentifiedObject

Table 55 shows all association ends of Asset with other classes.

Table 55 – Association ends of Assets::Asset with other classes

[mult from]	[mult to] name	type	description
[0..*]	[0..*] ActivityRecords	ActivityRecord	All activity records created for this asset.

6.2.4.6 AssetContainer

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

Table 56 shows all attributes of AssetContainer.

Table 56 – Attributes of Assets::AssetContainer

name	type	description
category	String	inherited from: Asset
corporateCode	String	inherited from: Asset
utcNumber	String	inherited from: Asset
serialNumber	String	inherited from: Asset
lotNumber	String	inherited from: Asset
manufacturedDate	AbsoluteDate	inherited from: Asset
purchasePrice	Money	inherited from: Asset
critical	Boolean	inherited from: Asset
application	String	inherited from: Asset
installationDate	AbsoluteDate	inherited from: Asset
acceptanceTest	AcceptanceTest	inherited from: Asset
initialCondition	String	inherited from: Asset
initialLossOfLife	PerCent	inherited from: Asset
status	Status	inherited from: Asset
electronicAddress	ElectronicAddress	inherited from: Asset
aliasName	String	inherited from: IdentifiedObject
description	String	inherited from: IdentifiedObject
localName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject
pathName	String	inherited from: IdentifiedObject

Table 57 shows all association ends of AssetContainer with other classes.

Table 57 – 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..*]	[0..*] ActivityRecords	ActivityRecord	inherited from: Asset

6.2.4.7 AssetFunction

Function performed by an asset.

Table 58 shows all attributes of AssetFunction.

Table 58 – 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
description	String	inherited from: IdentifiedObject
localName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject
pathName	String	inherited from: IdentifiedObject

6.2.4.8 ComMediaAsset

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

Table 59 shows all attributes of ComMediaAsset.

Table 59 – Attributes of Assets::ComMediaAsset

name	type	description
category	String	inherited from: Asset
corporateCode	String	inherited from: Asset
utcNumber	String	inherited from: Asset
serialNumber	String	inherited from: Asset
lotNumber	String	inherited from: Asset
manufacturedDate	AbsoluteDate	inherited from: Asset
purchasePrice	Money	inherited from: Asset
critical	Boolean	inherited from: Asset
application	String	inherited from: Asset
installationDate	AbsoluteDate	inherited from: Asset
acceptanceTest	AcceptanceTest	inherited from: Asset
initialCondition	String	inherited from: Asset
initialLossOfLife	PerCent	inherited from: Asset
status	Status	inherited from: Asset
electronicAddress	ElectronicAddress	inherited from: Asset
aliasName	String	inherited from: IdentifiedObject
description	String	inherited from: IdentifiedObject
localName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject
pathName	String	inherited from: IdentifiedObject

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

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

[mult from]	[mult to] name	type	description
[0..*]	[0..*] ActivityRecords	ActivityRecord	inherited from: Asset

6.2.4.9 Seal

Physically controls access to AssetContainers.

Table 61 shows all attributes of Seal.

Table 61 – Attributes of Assets::Seal

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

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

Table 62 – 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.

6.2.5 Package AssetModels

6.2.5.1 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 documentation classes describing assets of a particular product model made by a manufacturer. There are typically many instances of an asset associated with a single asset model. It also contains "lightweight" *Info classes, which hold model attributes that can be referenced by not only Assets but also by ConductingEquipments.

Figure 35 shows logical diagram AssetModelsInheritance.

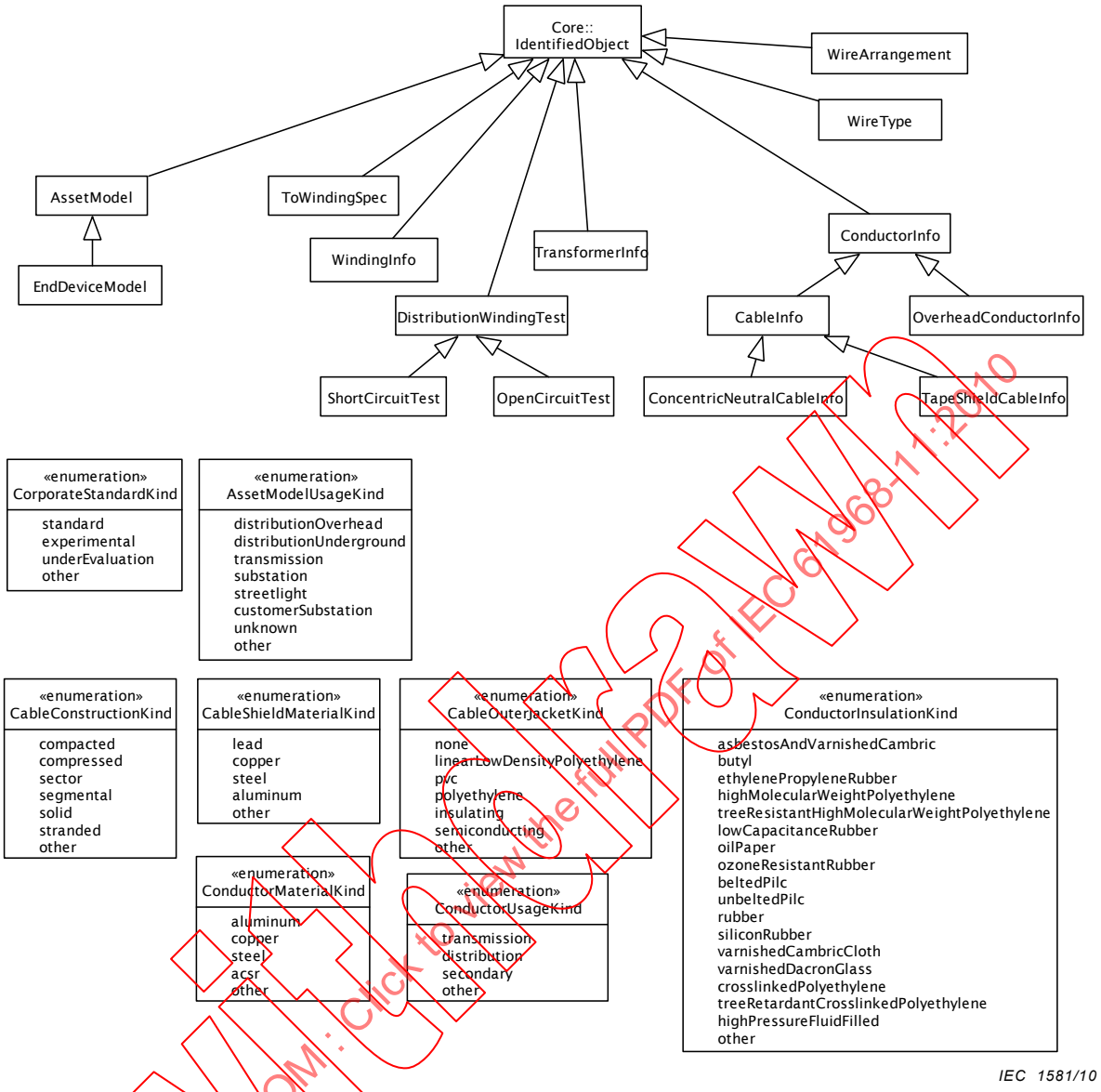


Figure 35 – Logical diagram AssetModels::AssetModelsInheritance

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

Figure 36 shows logical diagram AssetModelsOverview.

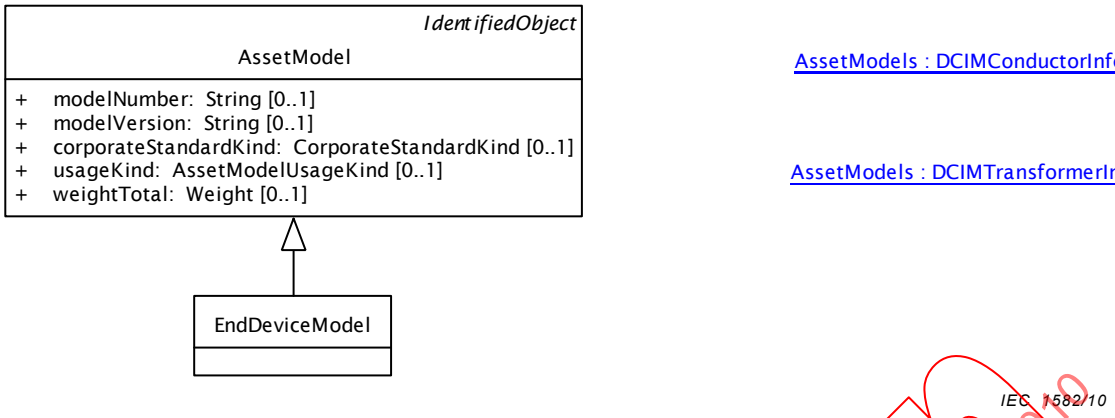


Figure 36 – Logical diagram AssetModels::AssetModelsOverview

This diagram shows normative classes from this package.

Figure 37 shows logical diagram DCIMConductorInfo.

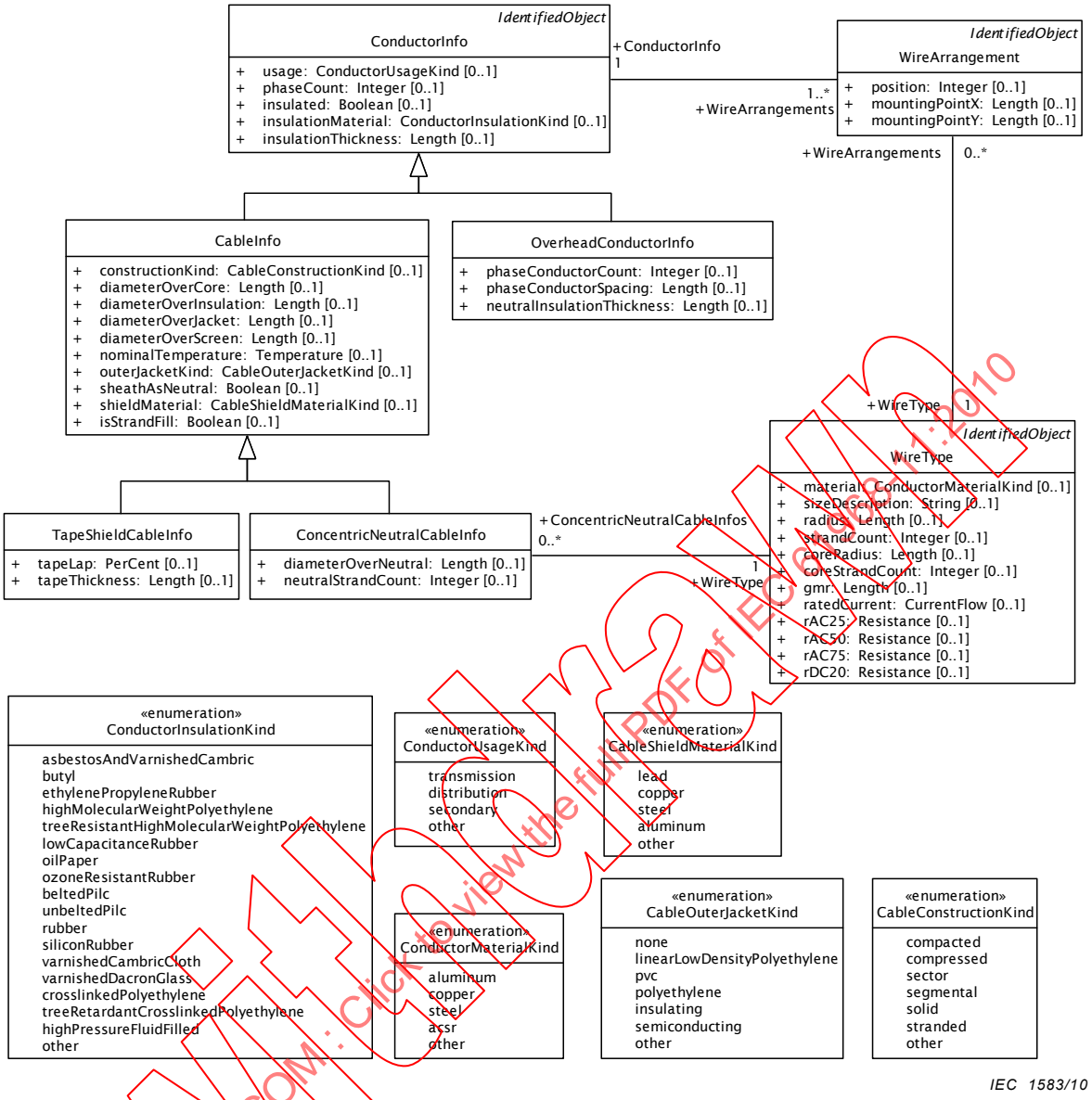


Figure 37 – Logical diagram AssetModels::DCIMConductorInfo

This diagram shows classes used to model conductors in DCIM.

Figure 38 shows logical diagram DCIMTransformerInfo.

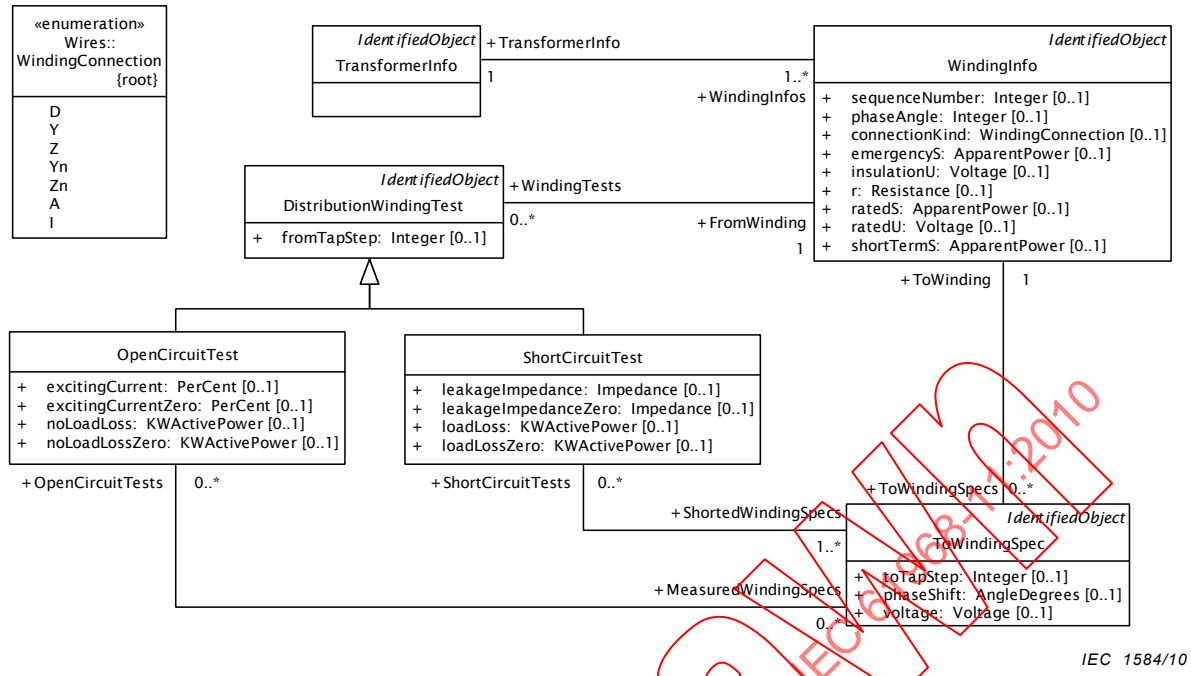


Figure 38 – Logical diagram AssetModels::DCIMTransformerInfo

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

6.2.5.2 AssetModelUsageKind enumeration

Usage for an asset model.

Table 63 shows all literals of AssetModelUsageKind.

Table 63 – Literals of AssetModels::AssetModelUsageKind

literal	description
distributionOverhead	
distributionUnderground	
transmission	
substation	
streetlight	
customerSubstation	
unknown	
other	

6.2.5.3 CorporateStandardKind enumeration

Kind of corporate standard.

Table 64 shows all literals of CorporateStandardKind.

Table 64 – Literals of AssetModels::CorporateStandardKind

literal	description
standard	
experimental	
underEvaluation	
other	

6.2.5.4 CableConstructionKind enumeration

Kind of cable construction.

Table 65 shows all literals of CableConstructionKind.

Table 65 – Literals of AssetModels::CableConstructionKind

literal	description
compacted	
compressed	
sector	
segmental	
solid	
stranded	
other	

6.2.5.5 CableOuterJacketKind enumeration

Kind of cable outer jacket.

Table 66 shows all literals of CableOuterJacketKind.

Table 66 – Literals of AssetModels::CableOuterJacketKind

literal	description
none	
linearLowDensityPolyethylene	
pvc	
polyethylene	
insulating	
semiconducting	
other	

6.2.5.6 CableShieldMaterialKind enumeration

Kind of cable shield material.

Table 67 shows all literals of CableShieldMaterialKind.

Table 67 – Literals of AssetModels::CableShieldMaterialKind

literal	description
lead	
copper	
steel	
aluminum	
other	

6.2.5.7 ConductorInsulationKind enumeration

Kind of conductor insulation.

Table 68 shows all literals of ConductorInsulationKind.

Table 68 – Literals of AssetModels::ConductorInsulationKind

literal	description
asbestosAndVarnishedCambric	
butyl	
ethylenePropyleneRubber	
highMolecularWeightPolyethylene	
treeResistantHighMolecularWeightPolyethylene	
lowCapacitanceRubber	
oilPaper	
ozoneResistantRubber	
beltedPilc	
unbeltedPilc	
rubber	
siliconRubber	
varnishedCambricCloth	
varnishedDacronGlass	
crosslinkedPolyethylene	
treeRetardantCrosslinkedPolyethylene	
highPressureFluidFilled	
other	

6.2.5.8 ConductorMaterialKind enumeration

Kind of conductor material.

Table 69 shows all literals of ConductorMaterialKind.

Table 69 – Literals of AssetModels::ConductorMaterialKind

literal	description
aluminum	
copper	
steel	
acsr	Aluminum conductor steel reinforced.
other	

6.2.5.9 ConductorUsageKind enumeration

Kind of conductor usage.

Table 70 shows all literals of ConductorUsageKind.

Table 70 – Literals of AssetModels::ConductorUsageKind

literal	description
transmission	
distribution	
secondary	
other	

6.2.5.10 AssetModel

Documentation for a particular product model made by a manufacturer. There are typically many instances of an asset associated with a single asset model.

Table 71 shows all attributes of AssetModel.

Table 71 – Attributes of AssetModels::AssetModel

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
description	String	inherited from: IdentifiedObject
localName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject

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

6.2.5.11 CableInfo

Cable data.

Table 72 shows all attributes of CableInfo.

Table 72 – Attributes of AssetModels::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.
usage	ConductorUsageKind	inherited from: ConductorInfo
phaseCount	Integer	inherited from: ConductorInfo
insulationMaterial	ConductorInsulationKind	inherited from: ConductorInfo
insulationThickness	Length	inherited from: ConductorInfo
insulated	Boolean	inherited from: ConductorInfo
aliasName	String	inherited from: IdentifiedObject
description	String	inherited from: IdentifiedObject
localName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject
pathName	String	inherited from: IdentifiedObject

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

Table 73 – Association ends of AssetModels::CableInfo with other classes

[mult from]	[mult to] name	type	description
[1..1]	[1..*] WireArrangements	WireArrangement	inherited from: ConductorInfo
[0..1]	[0..*] ConductorSegments	DistributionLineSegment	inherited from: ConductorInfo

6.2.5.12 ConcentricNeutralCableInfo

Concentric neutral cable data.

Table 74 shows all attributes of ConcentricNeutralCableInfo.

Table 74 – Attributes of AssetModels::ConcentricNeutralCableInfo

name	type	description
diameterOverNeutral	Length	Diameter over the concentric neutral strands.
neutralStrandCount	Integer	Number of concentric neutral strands.
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
usage	ConductorUsageKind	inherited from: ConductorInfo
phaseCount	Integer	inherited from: ConductorInfo
insulationMaterial	ConductorInsulationKind	inherited from: ConductorInfo
insulationThickness	Length	inherited from: ConductorInfo
insulated	Boolean	inherited from: ConductorInfo
aliasName	String	inherited from: IdentifiedObject
description	String	inherited from: IdentifiedObject
localName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject
pathName	String	inherited from: IdentifiedObject

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

Table 75 – Association ends of AssetModels::ConcentricNeutralCableInfo with other classes

[mult from]	[mult to] name	type	description
[0..*]	[1..1] WireType	WireType	Wire type used for this concentric neutral cable.
[1..1]	[1..*] WireArrangements	WireArrangement	inherited from: ConductorInfo
[0..1]	[0..*] ConductorSegments	DistributionLineSegment	inherited from: ConductorInfo

6.2.5.13 ConductorInfo

Conductor data.

Table 76 shows all attributes of ConductorInfo.

Table 76 – Attributes of AssetModels::ConductorInfo

name	type	description
usage	ConductorUsageKind	Usage of this conductor.
phaseCount	Integer	Number of phases (including neutral) to be retained. Any wires beyond this number should be reduced into the earth return.
insulationMaterial	ConductorInsulationKind	(if insulated conductor) Material used for insulation.
insulationThickness	Length	(if insulated conductor) Thickness of the insulation.
insulated	Boolean	True if conductor is insulated.
aliasName	String	inherited from: IdentifiedObject
description	String	inherited from: IdentifiedObject
localName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject
pathName	String	inherited from: IdentifiedObject

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

Table 77 – Association ends of AssetModels::ConductorInfo with other classes

[mult from]	[mult to] name	type	description
[1..1]	[1..*] WireArrangements	WireArrangement	All wire arrangements (single wires) that make this conductor.
[0..1]	[0..*] ConductorSegments	DistributionLineSegment	All conductor segments described by this conductor data.

6.2.5.14 DistributionWindingTest

Test results for one or more transformer windings. These may include short-circuit or open-circuit (excitation) tests. Short-circuit test results include load losses and leakage impedances. Open-circuit test results may include no-load losses, exciting current, phase shifts, and induced voltage. For three-phase windings, the excitation can be positive sequence (the default) or zero sequence.

Table 78 shows all attributes of DistributionWindingTest.

Table 78 – Attributes of AssetModels::DistributionWindingTest

name	type	description
fromTapStep	Integer	Tap step number for the "from" winding of the test pair.
aliasName	String	inherited from: IdentifiedObject
description	String	inherited from: IdentifiedObject
localName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject

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

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

Table 79 – Association ends of AssetModels::DistributionWindingTest with other classes

[mult from]	[mult to] name	type	description
[0..*]	[1..1] FromWinding	WindingInfo	Winding that voltage or current is applied to during the test.

6.2.5.15 EndDeviceModel

Documentation for particular end device product model made by a manufacturer.

Table 80 shows all attributes of EndDeviceModel.

Table 80 – Attributes of AssetModels::EndDeviceModel

name	type	description
modelNumber	String	inherited from: AssetModel
modelVersion	String	inherited from: AssetModel
corporateStandardKind	CorporateStandardKind	inherited from: AssetModel
usageKind	AssetModelUsageKind	inherited from: AssetModel
weightTotal	Weight	inherited from: AssetModel
aliasName	String	inherited from: IdentifiedObject
description	String	inherited from: IdentifiedObject
localName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject
pathName	String	inherited from: IdentifiedObject

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

Table 81 – Association ends of AssetModels::EndDeviceModel with other classes

[mult from]	[mult to] name	type	description
[0..1]	[0..*] EndDeviceAssets	EndDeviceAsset	All end device assets being of this model.

6.2.5.16 OpenCircuitTest

Open-circuit test results may include no-load losses, exciting current, phase shifts, and induced voltage. For three-phase windings, the excitation can be positive sequence (the default) or zero sequence.

For induced voltage and phase shifts, use the associated ToWindingSpec class.

Table 82 shows all attributes of OpenCircuitTest.

Table 82 – Attributes of AssetModels::OpenCircuitTest

name	type	description
excitingCurrent	PerCent	Exciting current measured from a positive-sequence or single-phase open-circuit (excitation) test.
excitingCurrentZero	PerCent	Exciting current measured from a zero-sequence open-circuit (excitation) test.
noLoadLoss	KWActivePower	Losses measured from a positive-sequence or single-phase open-circuit (excitation) test.
noLoadLossZero	KWActivePower	Losses measured from a zero-sequence open-circuit (excitation) test.
fromTapStep	Integer	inherited from: DistributionWindingTest
aliasName	String	inherited from: IdentifiedObject
description	String	inherited from: IdentifiedObject
localName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject
pathName	String	inherited from: IdentifiedObject

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

Table 83 – Association ends of AssetModels::OpenCircuitTest with other classes

[mult from]	[mult to] name	type	description
[0..*]	[0..*] MeasuredWindingSpecs	ToWindingSpec	All other windings measured during this test.
[0..*]	[1..1] FromWinding	WindingInfo	inherited from: DistributionWindingTest

6.2.5.17 OverheadConductorInfo

Overhead conductor data.

Table 84 shows all attributes of OverheadConductorInfo.

Table 84 – Attributes of AssetModels::OverheadConductorInfo

name	type	description
phaseConductorCount	Integer	Number of conductor strands in the symmetrical bundle (1-12).
phaseConductorSpacing	Length	Distance between conductor strands in a symmetrical bundle.
neutralInsulationThickness	Length	(if applicable) Insulation thickness of the neutral conductor.
usage	ConductorUsageKind	inherited from: ConductorInfo
phaseCount	Integer	inherited from: ConductorInfo

name	type	description
insulationMaterial	ConductorInsulationKind	inherited from: ConductorInfo
insulationThickness	Length	inherited from: ConductorInfo
insulated	Boolean	inherited from: ConductorInfo
aliasName	String	inherited from: IdentifiedObject
description	String	inherited from: IdentifiedObject
localName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject
pathName	String	inherited from: IdentifiedObject

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

Table 85 – Association ends of AssetModels::OverheadConductorInfo with other classes

[mult from]	[mult to] name	type	description
[1..1]	[1..*] WireArrangements	WireArrangement	inherited from: ConductorInfo
[0..1]	[0..*] ConductorSegments	DistributionLineSegment	inherited from: ConductorInfo

6.2.5.18 ShortCircuitTest

Short-circuit test results include load losses and leakage impedances. For three-phase windings, the excitation can be positive sequence (the default) or zero sequence. There must be at least one short-circuited ("to") winding.

Table 86 shows all attributes of ShortCircuitTest.

Table 86 – Attributes of AssetModels::ShortCircuitTest

name	type	description
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.
loadLoss	KWActivePower	Load losses from a positive-sequence or single-phase short-circuit test.
loadLossZero	KWActivePower	Load losses from a zero-sequence short-circuit test.
fromTapStep	Integer	inherited from: DistributionWindingTest
aliasName	String	inherited from: IdentifiedObject
description	String	inherited from: IdentifiedObject
localName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject
pathName	String	inherited from: IdentifiedObject

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

Table 87 – Association ends of AssetModels::ShortCircuitTest with other classes

[mult from]	[mult to] name	type	description
[0..*]	[1..*] ShortedWindingSpecs	ToWindingSpec	All windings short-circuited during this test.
[0..*]	[1..1] FromWinding	WindingInfo	inherited from: DistributionWindingTest

6.2.5.19 TapeShieldCableInfo

Tape shield cable data.

Table 88 shows all attributes of TapeShieldCableInfo.

Table 88 – Attributes of AssetModels::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
usage	ConductorUsageKind	inherited from: ConductorInfo
phaseCount	Integer	inherited from: ConductorInfo
insulationMaterial	ConductorInsulationKind	inherited from: ConductorInfo
insulationThickness	Length	inherited from: ConductorInfo
insulated	Boolean	inherited from: ConductorInfo
aliasName	String	inherited from: IdentifiedObject
description	String	inherited from: IdentifiedObject
localName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject
pathName	String	inherited from: IdentifiedObject

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

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

[mult from]	[mult to] name	type	description
[1..1]	[1..*] WireArrangements	WireArrangement	inherited from: ConductorInfo
[0..1]	[0..*] ConductorSegments	DistributionLineSegment	inherited from: ConductorInfo

6.2.5.20 ToWindingSpec

For short-circuit tests, specifies the winding and tap for all short-circuited windings.

For open-circuit tests, specifies the winding, tap, induced voltage, and induced angle for any non-excited windings that were measured during the test. This won't apply if only the exciting current and no-load losses were measured.

Table 90 shows all attributes of ToWindingSpec.

Table 90 – Attributes of AssetModels::ToWindingSpec

name	type	description
toTapStep	Integer	Tap step number for the "to" winding of the test pair.
phaseShift	AngleDegrees	(if open-circuit test) Phase shift measured at the open-circuited "to" winding, with the "from" winding set to the "from" winding's rated voltage and all other windings open-circuited.
voltage	Voltage	(if open-circuit test) Voltage measured at the open-circuited "to" winding, with the "from" winding set to the "from" winding's rated voltage and all other windings open-circuited.
aliasName	String	inherited from: IdentifiedObject
description	String	inherited from: IdentifiedObject
localName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject
pathName	String	inherited from: IdentifiedObject

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

Table 91 – Association ends of AssetModels::ToWindingSpec with other classes

[mult from]	[mult to] name	type	description
[0..*]	[1..1] ToWinding	WindingInfo	Winding short-circuited in a short-circuit test, or measured for induced voltage and angle in an open-circuit test.
[0..*]	[0..*] OpenCircuitTests	OpenCircuitTest	All open-circuit tests in which this winding was measured.
[1..*]	[0..*] ShortCircuitTests	ShortCircuitTest	All short-circuit tests in which this winding was short-circuited.

6.2.5.21 TransformerInfo

Set of transformer data, from an equipment library.

Table 92 shows all attributes of TransformerInfo.

Table 92 – Attributes of AssetModels::TransformerInfo

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

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

Table 93 – Association ends of AssetModels::TransformerInfo with other classes

[mult from]	[mult to] name	type	description
[1..1]	[1..*] WindingInfos	WindingInfo	Data for all the windings described by this transformer data.
[0..1]	[0..*] Transformers	DistributionTransformer	All transformers that can be described with this transformer data.

6.2.5.22 WindingInfo

Winding data.

Table 94 shows all attributes of WindingInfo.

Table 94 – Attributes of AssetModels::WindingInfo

name	type	description
phaseAngle	Integer	Winding phase angle where 360 degrees are represented with clock hours, so the valid values are {0, ..., 11}. For example, to express winding code 'Dyn11', set attributes as follows: 'connectionKind' = Yn and 'phaseAngle' = 11.
connectionKind	WindingConnection	Kind of connection of this winding.
emergencyS	ApparentPower	Apparent power that the winding can carry under emergency conditions.
insulationU	Voltage	Basic insulation level voltage rating.
r	Resistance	DC resistance of this winding.
ratedS	ApparentPower	Normal apparent power rating of this winding.
ratedU	Voltage	Rated voltage of this winding: phase-phase for three-phase windings, and either phase-phase or phase-neutral for single-phase windings.
shortTermS	ApparentPower	Apparent power that this winding can carry for a short period of time.

name	type	description
sequenceNumber	Integer	Sequence number for this winding, corresponding to the winding's order in the TransformerBank.vectorGroup attribute. Highest voltage winding should be '1'.
aliasName	String	inherited from: IdentifiedObject
description	String	inherited from: IdentifiedObject
localName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject
pathName	String	inherited from: IdentifiedObject

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

Table 95 – Association ends of AssetModels::WindingInfo with other classes

[mult from]	[mult to] name	type	description
[1..1]	[0..*] WindingTests	DistributionWindingTest	All winding tests during which voltage or current was applied to this winding.
[0..1]	[0..*] Windings	DistributionTransformerWinding	All windings described by this winding data.
[1..1]	[0..*] ToWindingSpecs	ToWindingSpec	Tap steps and induced voltage/angle measurements for tests in which this winding was not excited.
[1..*]	[1..1] TransformerInfo	TransformerInfo	Transformer data that this winding description is part of.

6.2.5.23 WireArrangement

Identification, spacing and configuration of the wires of a conductor, with reference to their type.

Table 96 shows all attributes of WireArrangement.

Table 96 – Attributes of AssetModels::WireArrangement

name	type	description
position	Integer	Position number on the structure corresponding to this wire. For example, use 1..3 for phases and 4 for the neutral on a 3-phase structure. The individual phase assignments matter; for example, ABC will produce a different set of unbalanced line parameters, by phase, than BAC.
mountingPointX	Length	Signed horizontal distance from the first wire to a common reference point.
mountingPointY	Length	Height above ground of the first wire.
aliasName	String	inherited from: IdentifiedObject
description	String	inherited from: IdentifiedObject
localName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject

name	type	description
pathName	String	inherited from: IdentifiedObject

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

Table 97 – Association ends of AssetModels::WireArrangement with other classes

[mult from]	[mult to] name	type	description
[0..*]	[1..1] WireType	WireType	Wire type used for this wire arrangement.
[1..*]	[1..1] ConductorInfo	ConductorInfo	Conductor data this wire arrangement belongs to.

6.2.5.24 WireType

Wire conductor (per IEEE specs). A specific type of wire or combination of wires, not insulated from each other, suitable for carrying electrical current.

Table 98 shows all attributes of WireType.

Table 98 – Attributes of AssetModels::WireType

name	type	description
material	ConductorMaterialKind	Wire material.
sizeDescription	String	Describes the wire guage or cross section (e.g., 4/0, #2, 336.5).
radius	Length	Outside radius of the wire.
strandCount	Integer	Number of strands in the wire.
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	Resistance	AC resistance per unit length of the conductor at 25 ˚C.
rAC50	Resistance	AC resistance per unit length of the conductor at 50 ˚C.
rAC75	Resistance	AC resistance per unit length of the conductor at 75 ˚C.
rDC20	Resistance	DC resistance per unit length of the conductor at 20 ˚C.
aliasName	String	inherited from: IdentifiedObject
description	String	inherited from: IdentifiedObject
localName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject
pathName	String	inherited from: IdentifiedObject

Table 99 shows all association ends of WireType with other classes.

Table 99 – Association ends of AssetModels::WireType with other classes

[mult from]	[mult to] name	type	description
[1..1]	[0..*] ConcentricNeutralCableInfos	ConcentricNeutralCableInfo	All concentric neutral cables using this wire type.
[1..1]	[0..*] WireArrangements	WireArrangement	All wire arrangements using this wire type.

6.2.6 Package Work

6.2.6.1 General

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

Figure 39 shows logical diagram WorkInheritance.

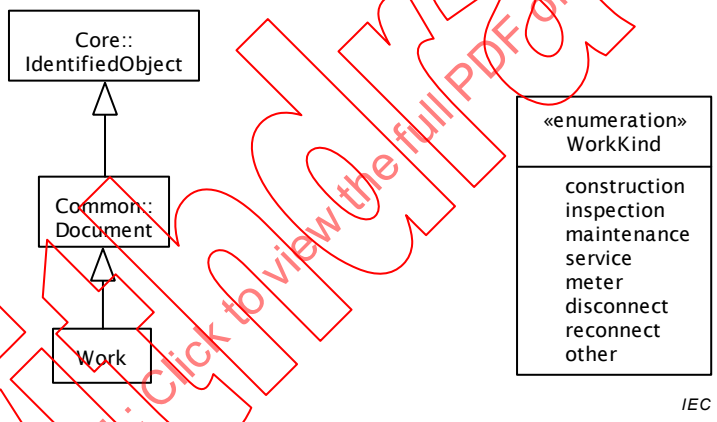


Figure 39 – Logical diagram Work::WorkInheritance

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

Figure 40 shows logical diagram WorkOverview.

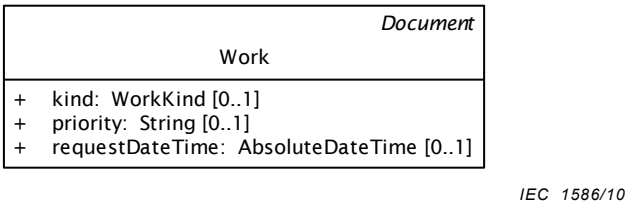


Figure 40 – Logical diagram Work::WorkOverview

This diagram shows normative classes from this package.

6.2.6.2 WorkKind enumeration

Kind of work.

Table 100 shows all literals of WorkKind.

Table 100 – Literals of Work::WorkKind

literal	description
construction	
inspection	
maintenance	
service	
meter	
disconnect	
reconnect	
other	

6.2.6.3 Work

Document used to request, initiate, track and record work. This is synonymous with work breakdown structure (WBS), which is traversed through the (currently informative) recursive association of work.

Note that the work name is equal to the WBS name, which is given in the inherited "name" attribute.

Table 101 shows all attributes of Work.

Table 101 – Attributes of Work::Work

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

Table 102 shows all association ends of Work with other classes.

Table 102 – 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..*]	[0..*] ActivityRecords	ActivityRecord	inherited from: Document

6.2.7 Package Customers

6.2.7.1 General

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

Figure 41 shows logical diagram CustomersInheritance.

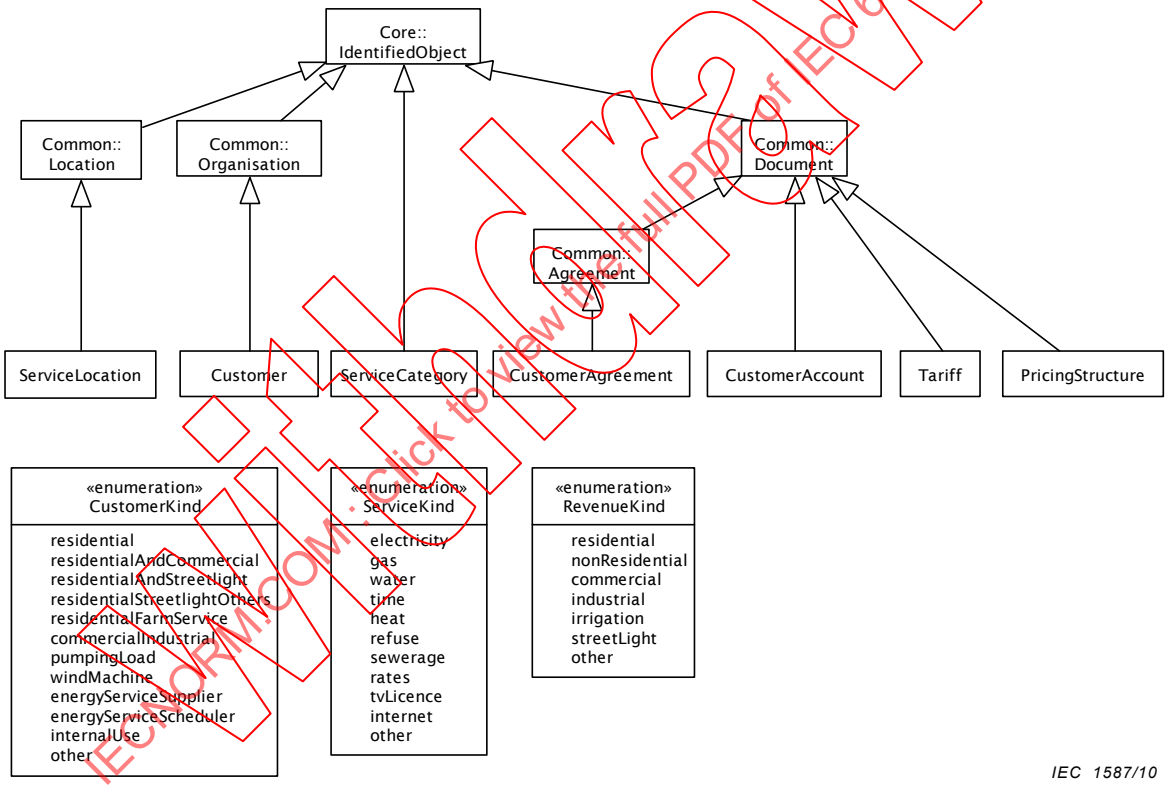


Figure 41 – Logical diagram Customers::CustomersInheritance

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

Figure 42 shows logical diagram CustomersOverview.

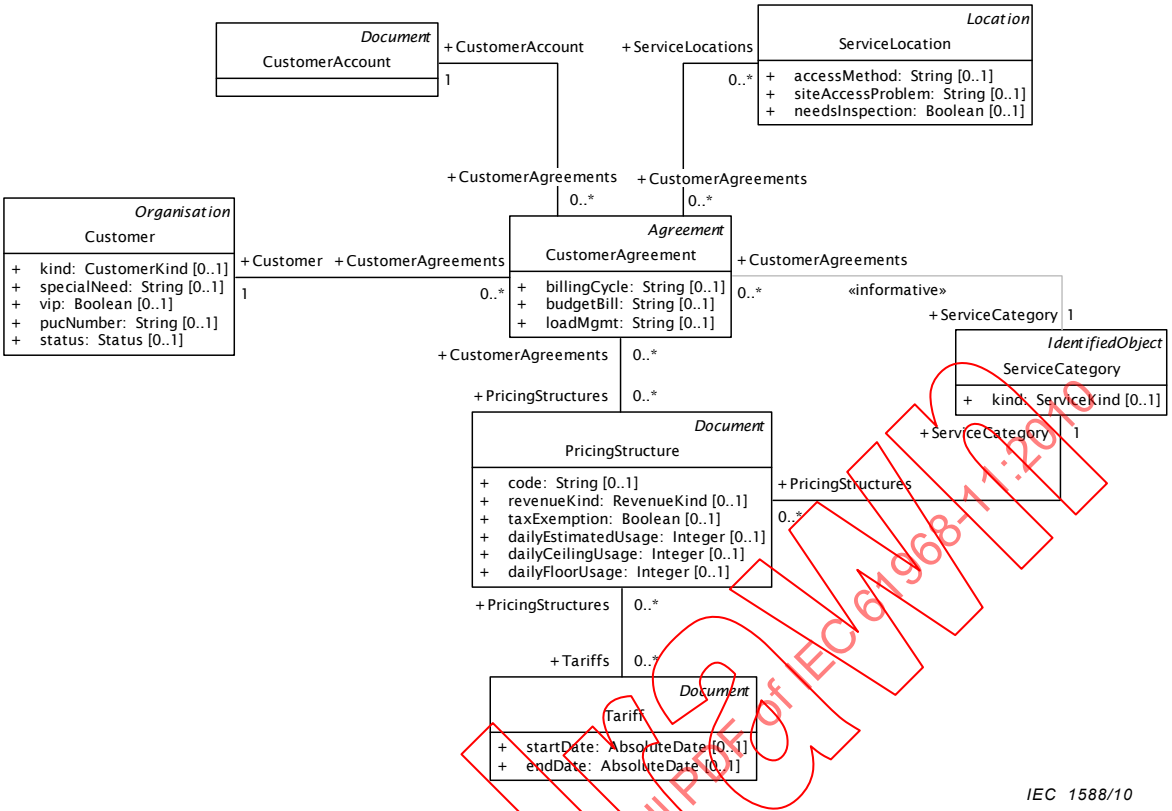


Figure 42 – Logical diagram Customers::CustomersOverview

This diagram shows normative classes from this package.

6.2.7.2 CustomerKind enumeration

Kind of customer.

Table 103 shows all literals of CustomerKind.

Table 103 – Literals of Customers::CustomerKind

literal	description
residential	
residentialAndCommercial	
residentialAndStreetlight	
residentialStreetlightOthers	
residentialFarmService	
commercialIndustrial	
pumpingLoad	
windMachine	
energyServiceSupplier	
energyServiceScheduler	
internalUse	
other	

6.2.7.3 RevenueKind enumeration

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

Table 104 shows all literals of RevenueKind.

Table 104 – Literals of Customers::RevenueKind

literal	description
residential	
nonResidential	
commercial	
industrial	
irrigation	
streetLight	
other	

6.2.7.4 ServiceKind enumeration

Kind of service.

Table 105 shows all literals of ServiceKind.

Table 105 – Literals of Customers::ServiceKind

literal	description
electricity	
gas	
water	
time	
heat	
refuse	
sewerage	
rates	
tvLicence	
internet	
other	

6.2.7.5 Customer

Organisation receiving services from ServiceSupplier.

Table 106 shows all attributes of Customer.

Table 106 – Attributes of Customers::Customer

name	type	description
kind	CustomerKind	Kind of customer.
specialNeed	String	True if customer organisation has special service

name	type	description
		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 utility commission identification number.
status	Status	Status of this customer.
streetAddress	StreetAddress	inherited from: Organisation
postalAddress	PostalAddress	inherited from: Organisation
phone1	TelephoneNumber	inherited from: Organisation
phone2	TelephoneNumber	inherited from: Organisation
electronicAddress	ElectronicAddress	inherited from: Organisation
aliasName	String	inherited from: IdentifiedObject
description	String	inherited from: IdentifiedObject
localName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject
pathName	String	inherited from: IdentifiedObject

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

Table 107 – Association ends of Customers::Customer with other classes

[mult from]	[mult to] name	type	description
[0..1]	[0..*] EndDeviceAssets	EndDeviceAsset	All end device assets of this customer.
[1..1]	[0..*] CustomerAgreements	CustomerAgreement	All agreements of this customer.
[0..*]	[0..*] Works	Work	All the works performed for this customer.

6.2.7.6 CustomerAccount

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

Table 108 shows all attributes of CustomerAccount.

Table 108 – Attributes of Customers::CustomerAccount

name	type	description
category	String	inherited from: Document
createdDateTime	AbsoluteDateTime	inherited from: Document
lastModifiedDateTime	AbsoluteDateTime	inherited from: Document
revisionNumber	String	inherited from: Document
subject	String	inherited from: Document
title	String	inherited from: Document

name	type	description
docStatus	Status	inherited from: Document
status	Status	inherited from: Document
electronicAddress	ElectronicAddress	inherited from: Document
aliasName	String	inherited from: IdentifiedObject
description	String	inherited from: IdentifiedObject
localName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject
pathName	String	inherited from: IdentifiedObject

Table 109 shows all association ends of CustomerAccount with other classes.

Table 109 – 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..*]	[0..*] ActivityRecords	ActivityRecord	inherited from: Document

6.2.7.7 CustomerAgreement

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

Table 110 shows all attributes of CustomerAgreement.

Table 110 – Attributes of Customers::CustomerAgreement

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.
loadMgmt	String	Load management code.
signDate	AbsoluteDate	inherited from: Agreement
validityInterval	DateTimeInterval	inherited from: Agreement
category	String	inherited from: Document
createdDateTime	AbsoluteDateTime	inherited from: Document
lastModifiedDateTime	AbsoluteDateTime	inherited from: Document
revisionNumber	String	inherited from: Document
subject	String	inherited from: Document
title	String	inherited from: Document
docStatus	Status	inherited from: Document

name	type	description
status	Status	inherited from: Document
electronicAddress	ElectronicAddress	inherited from: Document
aliasName	String	inherited from: IdentifiedObject
description	String	inherited from: IdentifiedObject
localName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject
pathName	String	inherited from: IdentifiedObject

Table 111 shows all association ends of CustomerAgreement with other classes.

Table 111 – 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..*] ServiceDeliveryPoints	ServiceDeliveryPoint	All service delivery points regulated by this customer agreement.
[0..1]	[0..*] MeterReadings	MeterReading	(could be deprecated in the future) All meter readings for this customer agreement.
[0..*]	[1..1] ServiceSupplier	ServiceSupplier	Service supplier for this customer agreement.
[0..*]	[1..1] CustomerAccount	CustomerAccount	Customer account owning this agreement.
[0..*]	[1..1] Customer	Customer	Customer for this agreement.
[0..*]	[0..*] ActivityRecords	ActivityRecord	inherited from: Document

6.2.7.8 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 112 shows all attributes of PricingStructure.

Table 112 – 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

name	type	description
		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.
category	String	inherited from: Document
createdDateTime	AbsoluteDateTime	inherited from: Document
lastModifiedDateTime	AbsoluteDateTime	inherited from: Document
revisionNumber	String	inherited from: Document
subject	String	inherited from: Document
title	String	inherited from: Document
docStatus	Status	inherited from: Document
status	Status	inherited from: Document
electronicAddress	ElectronicAddress	inherited from: Document
aliasName	String	inherited from: IdentifiedObject
description	String	inherited from: IdentifiedObject
localName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject
pathName	String	inherited from: IdentifiedObject

Table 113 shows all association ends of PricingStructure with other classes.

Table 113 – Association ends of Customers::PricingStructure with other classes

[mult from]	[mult to] name	type	description
[0..*]	[0..*] ServiceDeliveryPoints	ServiceDeliveryPoint	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..*]	[0..*] ActivityRecords	ActivityRecord	inherited from: Document

6.2.7.9 ServiceCategory

Category of service provided to the customer.

Table 114 shows all attributes of ServiceCategory.

Table 114 – Attributes of Customers::ServiceCategory

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

Table 115 shows all association ends of ServiceCategory with other classes.

Table 115 – Association ends of Customers::ServiceCategory with other classes

[mult from]	[mult to] name	type	description
[1..1]	[0..*] PricingStructures	PricingStructure	All pricing structures applicable to this service category.
[0..1]	[0..*] ServiceDeliveryPoints	ServiceDeliveryPoint	All service delivery points that deliver this category of service.

6.2.7.10 ServiceLocation

A customer ServiceLocation has one or more ServiceDeliveryPoint(s), which in turn relate to meters. The location may be a point or a polygon, depending on the specific circumstances.

For distribution, the ServiceLocation is typically the location of the utility customer's premise. Because a customer's premise may have one or more meters, the ServiceDeliveryPoint is used to define the actual conducting equipment that the EndDeviceAsset attaches to at the utility customer's ServiceLocation.

For transmission, it is the point(s) of interconnection on the transmission provider's transmission system where capacity and/or energy transmitted by the transmission provider is made available to the receiving party.

Table 116 shows all attributes of ServiceLocation.

Table 116 – Attributes of Customers::ServiceLocation

name	type	description
accessMethod	String	Method for the service person to access the appropriate service locations. 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 site. Examples

name	type	description
		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.
category	String	inherited from: Location
corporateCode	String	inherited from: Location
mainAddress	StreetAddress	inherited from: Location
secondaryAddress	StreetAddress	inherited from: Location
direction	String	inherited from: Location
geoInfoReference	String	inherited from: Location
status	Status	inherited from: Location
phone1	TelephoneNumber	inherited from: Location
phone2	TelephoneNumber	inherited from: Location
electronicAddress	ElectronicAddress	inherited from: Location
aliasName	String	inherited from: IdentifiedObject
description	String	inherited from: IdentifiedObject
localName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject
pathName	String	inherited from: IdentifiedObject

Table 117 shows all association ends of ServiceLocation with other classes.

Table 117 – Association ends of Customers::ServiceLocation with other classes

[mult from]	[mult to] name	type	description
[0..1]	[0..*] EndDeviceAssets	EndDeviceAsset	All end device assets that measure the service delivered to this service location.
[0..1]	[0..*] ServiceDeliveryPoints	ServiceDeliveryPoint	All service delivery 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
[1..1]	[0..*] CoordinateSystems	CoordinateSystem	inherited from: Location

6.2.7.11 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.

Table 118 shows all attributes of Tariff.

Table 118 – Attributes of Customers::Tariff

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

Table 119 shows all association ends of Tariff with other classes.

Table 119 – 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..*]	[0..*] ActivityRecords	ActivityRecord	inherited from: Document

6.2.8 Package Metering

6.2.8.1 General

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

Figure 43 shows logical diagram MeteringInheritance.

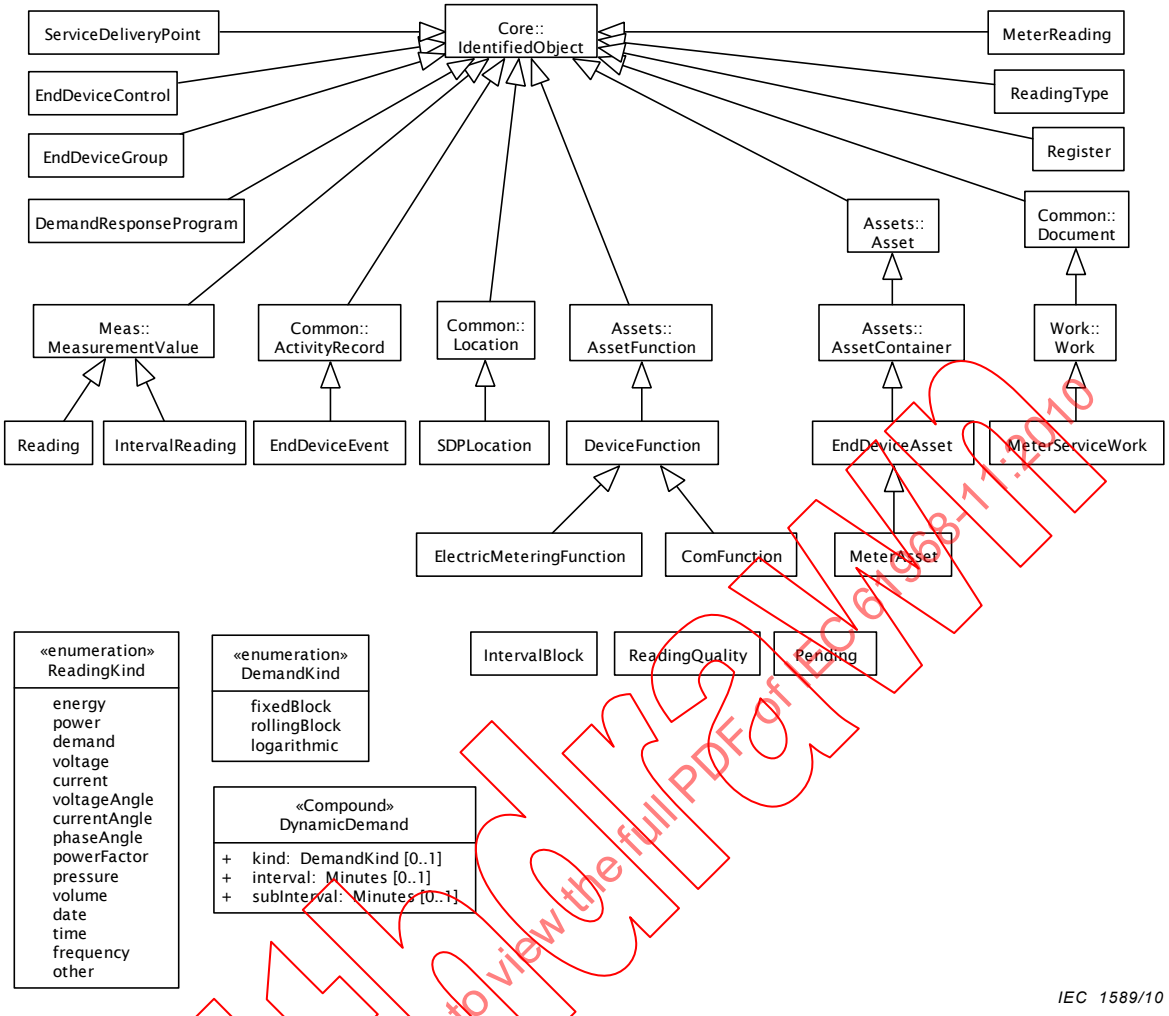


Figure 43 – Logical diagram Metering::MeteringInheritance

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

Figure 44 shows logical diagram MeteringOverviewShort.

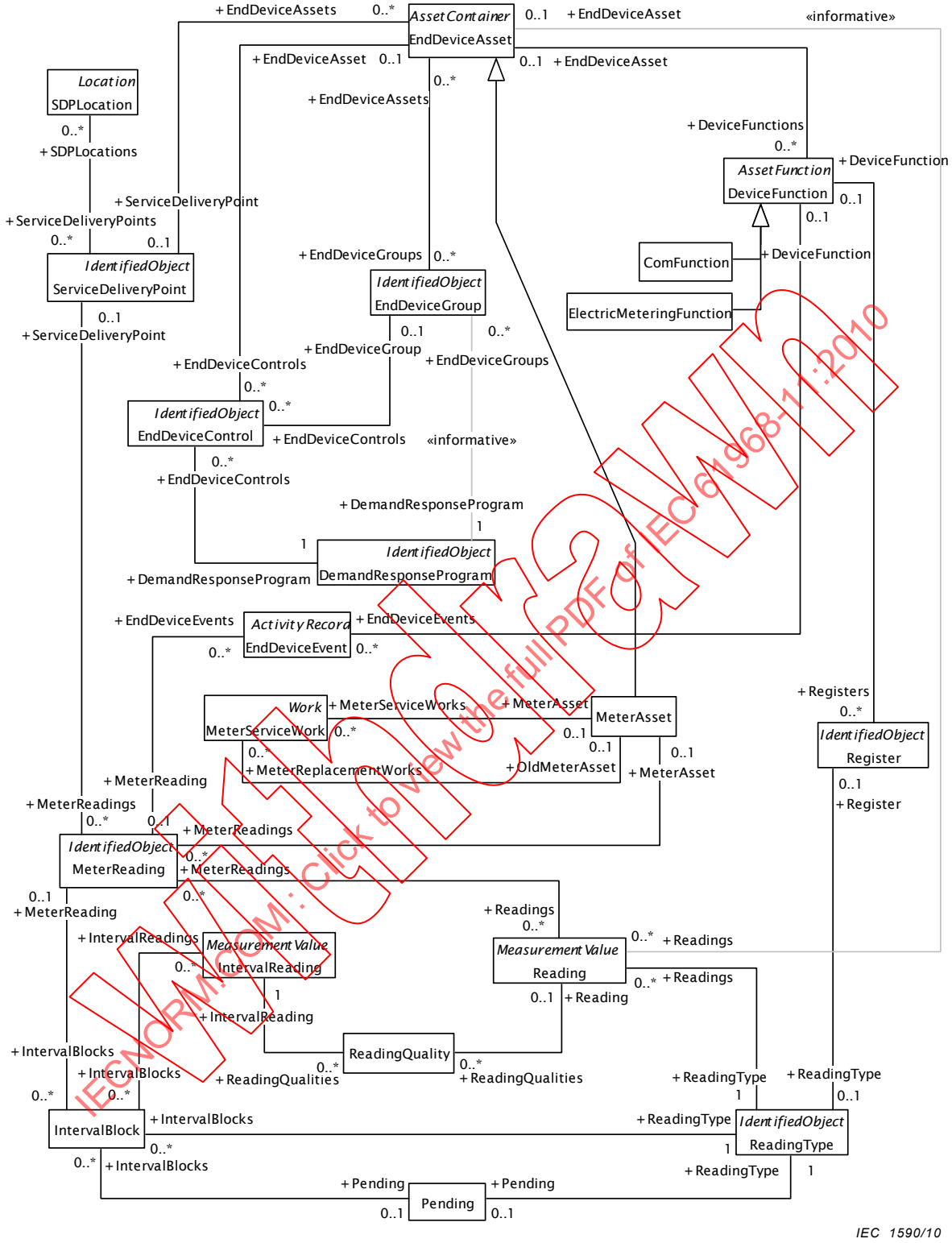


Figure 44 – Logical diagram Metering::MeteringOverviewShort

This diagram shows relationships among normative classes from this package (without their attributes).

Figure 45 shows logical diagram MeteringOverview.

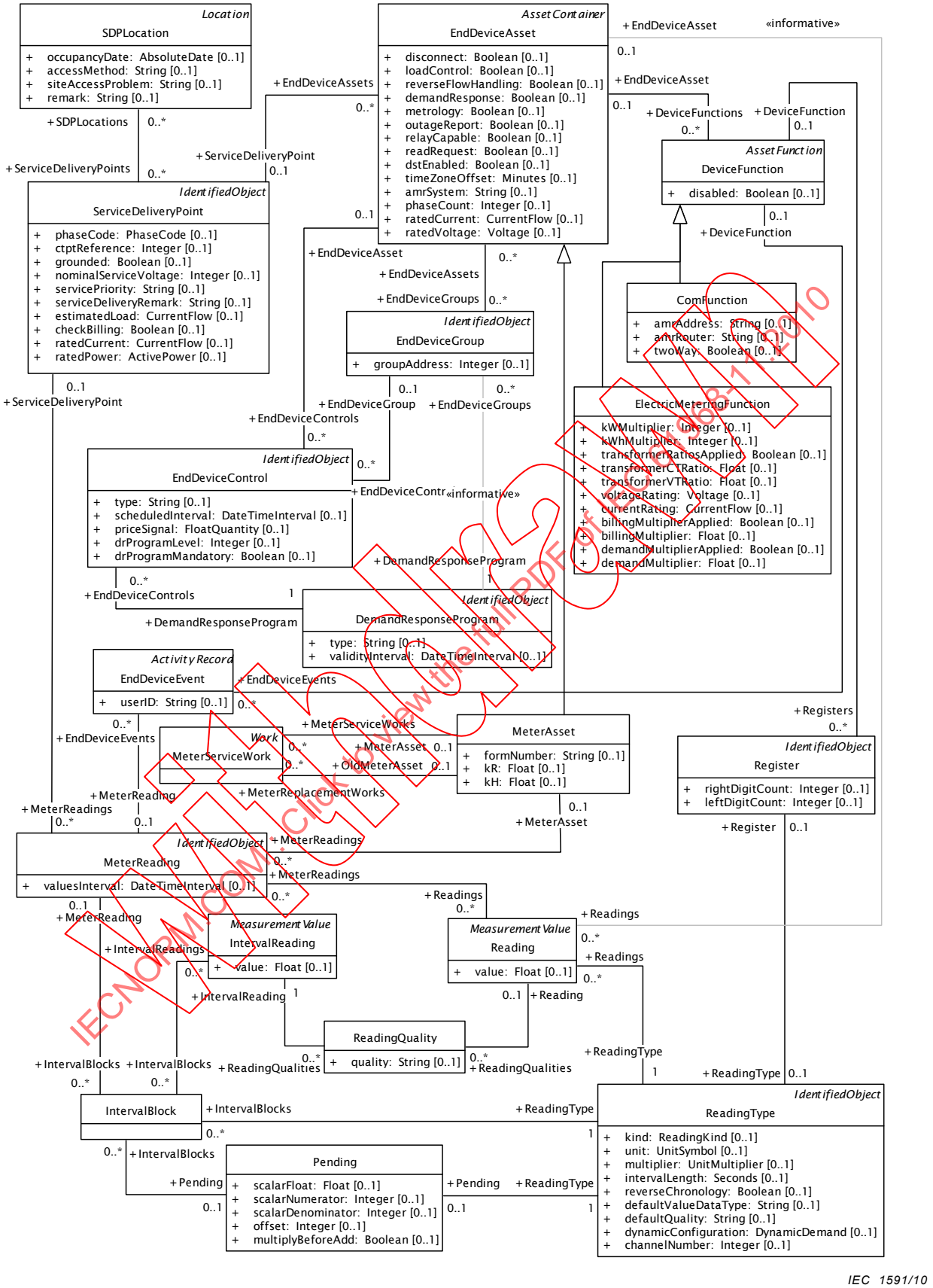


Figure 45 – Logical diagram Metering::MeteringOverview

This diagram shows normative classes from this package.

Figure 46 shows logical diagram MeteringRelationships.

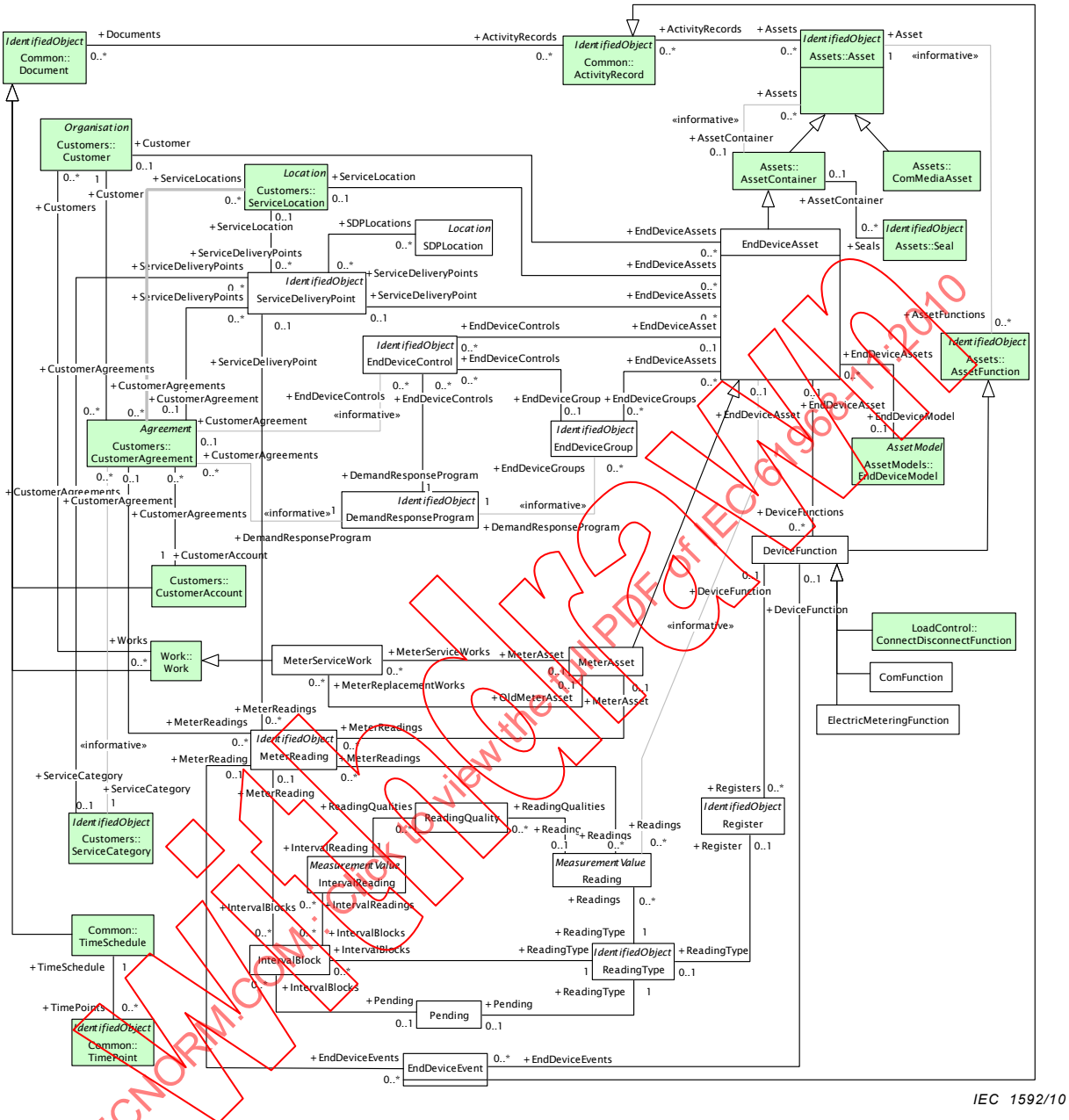


Figure 46 – Logical diagram Metering::MeteringRelationships

This diagram shows relationships of the classes from this package with those from other packages.

6.2.8.2 DemandKind enumeration

Kind of demand for dynamic meter configuration.

Table 120 shows all literals of DemandKind.

Table 120 – Literals of Metering::DemandKind

literal	description
fixedBlock	
rollingBlock	
logarithmic	

6.2.8.3 DynamicDemand compound

Dynamic demand description. The formula by which demand is measured is an important underlying definition to the measurement. Generally speaking, all of the meters in a given utility will be configured to measure demand the same way. Nevertheless, it must be defined. An 'interval' of 60 min, 30 min, 15 min, 10 min, or 5 min must be defined to describe the interval of time over which usage is measured. When demand is defined to be DemandKind.rollingBlock, both an 'interval' and a 'subinterval' must be defined, where the 'subinterval' must be a multiple of the 'interval' which contains it. A common setting is "15-minute rolling block with 5-minute subintervals."

Table 121 shows all attributes of DynamicDemand.

Table 121 – Attributes of Metering::DynamicDemand

name	type	description
kind	DemandKind	Kind of demand.
interval	Minutes	Demand interval.
subInterval	Minutes	(if 'kind'=rollingBlock) Subinterval, must be multiple of 'interval' that contains it.

6.2.8.4 ReadingKind enumeration

Kind of reading.

Table 122 shows all literals of ReadingKind.

Table 122 – Literals of Metering::ReadingKind

literal	description
energy	
power	
demand	
voltage	
current	
voltageAngle	
currentAngle	
phaseAngle	
powerFactor	
pressure	
volume	
date	

literal	description
time	
other	
frequency	

6.2.8.5 ComFunction

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

Table 123 shows all attributes of ComFunction.

Table 123 – 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. Note: If someone swaps out a meter, they may inadvertently disrupt the AMR system. Some technologies route readings from nearby meters through a common collection point on an electricity meter. Removal of such a meter disrupts AMR for numerous nearby meters.
twoWay	Boolean	True when the AMR module can both send and receive messages. Default is false (i.e., module can only send).
disabled	Boolean	inherited from: DeviceFunction
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
description	String	inherited from: IdentifiedObject
localName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject
pathName	String	inherited from: IdentifiedObject

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

Table 124 – Association ends of Metering::ComFunction with other classes

[mult from]	[mult to] name	type	description
[0..1]	[0..*] Registers	Register	inherited from: DeviceFunction
[0..*]	[0..1] EndDeviceAsset	EndDeviceAsset	inherited from: DeviceFunction
[0..1]	[0..*] EndDeviceEvents	EndDeviceEvent	inherited from: DeviceFunction

6.2.8.6 DemandResponseProgram

Demand response program.

Table 125 shows all attributes of DemandResponseProgram.

Table 125 – 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
description	String	inherited from: IdentifiedObject
localName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject
pathName	String	inherited from: IdentifiedObject

Table 126 shows all association ends of DemandResponseProgram with other classes.

Table 126 – Association ends of Metering::DemandResponseProgram with other classes

[mult from]	[mult to] name	type	description
[1..1]	[0..*] EndDeviceControls	EndDeviceControl	All end device controls with this demand response program.

6.2.8.7 DeviceFunction

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

Table 127 shows all attributes of DeviceFunction.

Table 127 – Attributes of Metering::DeviceFunction

name	type	description
disabled	Boolean	True if the device function is disabled (deactivated). Default is false (i.e., 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
description	String	inherited from: IdentifiedObject
localName	String	inherited from: IdentifiedObject

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

Table 128 shows all association ends of DeviceFunction with other classes.

Table 128 – Association ends of Metering::DeviceFunction with other classes

[mult from]	[mult to] name	type	description
[0..1]	[0..*] Registers	Register	All registers for quantities metered by this device function.
[0..*]	[0..1] EndDeviceAsset	EndDeviceAsset	End device asset that performs this function.
[0..1]	[0..*] EndDeviceEvents	EndDeviceEvent	All events reported by this device function.

6.2.8.8 ElectricMeteringFunction

Functionality performed by an electric meter.

Table 129 shows all attributes of ElectricMeteringFunction.

Table 129 – Attributes of Metering::ElectricMeteringFunction

name	type	description
kWMultiplier	Integer	Meter kW (pulse) multiplier, used as a multiplier for a meter register reading to determine the actual amount of usage for which to bill a customer.
kWhMultiplier	Integer	Meter kWh multiplier, used as a multiplier for a meter register reading to determine the actual amount of usage for which to bill a customer.
transformerRatiosApplied	Boolean	True if transformer ratios have been already applied to the associated quantities.
transformerCTRatio	Float	Current transformer ratio used to convert associated quantities to real measurements.
transformerVTRatio	Float	Voltage transformer ratio used to convert associated quantities to real measurements.
voltageRating	Voltage	The service voltage at which the meter is designed to operate. Typical voltage ratings in North America are 120 V, 240 V, 277 V or 480 V.
currentRating	CurrentFlow	The current class of the meter. Typical current classes in North America are 10 A, 20 A, 100 A, 200 A, or 320 A.
billingMultiplierApplied	Boolean	True if the billingMultiplier ratio has already been applied to the associated quantities.
billingMultiplier	Float	Customer billing value = meter multiplier * transformer ratios * reading value. The multiplier identifies the scaling value to apply to the reported value after delivery of the tagged item.
demandMultiplierApplied	Boolean	True if the demandMultiplier ratio has already been applied to the associated quantities.
demandMultiplier	Float	An additional multiplier that may be used for normalization of the demand value to an hourly value. For example, if the demand interval were set to 15 min, the demand multiplier would be 4.

name	type	description
		If the meter design is such that the demand value reported and displayed is compensated for by the meter itself and no additional scaling is required outside of the meter, the value of the demand multiplier should be "1".
disabled	Boolean	inherited from: DeviceFunction
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
description	String	inherited from: IdentifiedObject
localName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject
pathName	String	inherited from: IdentifiedObject

Table 130 shows all association ends of ElectricMeteringFunction with other classes.

Table 130 – Association ends of Metering::ElectricMeteringFunction with other classes

[mult from]	[mult to] name	type	description
[0..1]	[0..*] Registers	Register	inherited from: DeviceFunction
[0..*]	[0..1] EndDeviceAsset	EndDeviceAsset	inherited from: DeviceFunction
[0..1]	[0..*] EndDeviceEvents	EndDeviceEvent	inherited from: DeviceFunction

6.2.8.9 EndDeviceAsset

AssetContainer that performs one or more end device functions. One type of EndDeviceAsset is a MeterAsset which can perform metering, load management, connect/disconnect, accounting functions, etc. Some EndDeviceAssets, such as ones monitoring and controlling air conditioner, refrigerator, pool pumps may be connected to a MeterAsset. All EndDeviceAssets may have communication capability defined by the associated ComFunction(s). An EndDeviceAsset 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 131 shows all attributes of EndDeviceAsset.

Table 131 – Attributes of Metering::EndDeviceAsset

name	type	description
disconnect	Boolean	True if this end device asset is capable of supporting remote whole-house disconnect capability. To determine whether this functionality is installed and enabled, check the 'DeviceFunction.disabled' attribute of the

name	type	description
		ConnectDisconnectFunction contained by this end device asset. For example, to be able to remotely disconnect the device, the following values of attributes must hold: - EndDeviceAsset.disconnect = true (device supports disconnect) - ConnectDisconnectFunction.disabled (inherited from DeviceFunction) = false (function enabled) - ConnectDisconnectFunction.isConnected = true (currently connected).
loadControl	Boolean	True if this end device asset is capable of supporting load control functions through either the meter or the AMR option. To determine whether this functionality is installed and enabled, check the 'DeviceFunction.disabled' attribute of the respective function contained by this end device asset.
reverseFlowHandling	Boolean	True if this EndDeviceAsset is capable of supporting detection and monitoring of reverse flow.
demandResponse	Boolean	True if this end device asset is capable of supporting demand response functions. To determine whether this functionality is installed and enabled, check the 'DeviceFunction.disabled' attribute of the respective function contained by this end device asset.
metrology	Boolean	True if this end device asset is capable of supporting the presentation of metered values to a user or another system (always true for a meter, but might not be true for a load control unit.)
outageReport	Boolean	True if this end device asset is capable of supporting the means to report historical power interruption data.
relayCapable	Boolean	True if this end device asset is capable of supporting one or more relays. The relays may be programmable in the meter and tied to TOU, time pulse, load control or other functions. To determine whether this functionality is installed and enabled, check the 'DeviceFunction.disabled' attribute of the respective function contained by this end device asset.
readRequest	Boolean	True if this end device asset is capable of supporting on-request reads for this end device.
dstEnabled	Boolean	True if this end device asset is capable of supporting the autonomous application of daylight savings time (DST).
timeZoneOffset	Minutes	Time zone offset relative to GMT for the location of this end device.
amrSystem	String	Automated meter reading (AMR) system responsible for communications to this end device.
phaseCount	Integer	Number of potential phases the asset supports, typically 0, 1 or 3.
ratedCurrent	CurrentFlow	Rated current.
ratedVoltage	Voltage	Rated voltage.
category	String	inherited from: Asset
corporateCode	String	inherited from: Asset
utcNumber	String	inherited from: Asset

name	type	description
serialNumber	String	inherited from: Asset
lotNumber	String	inherited from: Asset
manufacturedDate	AbsoluteDate	inherited from: Asset
purchasePrice	Money	inherited from: Asset
critical	Boolean	inherited from: Asset
application	String	inherited from: Asset
installationDate	AbsoluteDate	inherited from: Asset
acceptanceTest	AcceptanceTest	inherited from: Asset
initialCondition	String	inherited from: Asset
initialLossOfLife	PerCent	inherited from: Asset
status	Status	inherited from: Asset
electronicAddress	ElectronicAddress	inherited from: Asset
aliasName	String	inherited from: IdentifiedObject
description	String	inherited from: IdentifiedObject
localName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject
pathName	String	inherited from: IdentifiedObject

Table 132 shows all association ends of EndDeviceAsset with other classes.

Table 132 – Association ends of Metering::EndDeviceAsset with other classes

[mult from]	[mult to] name	type	description
[0..1]	[0..*] DeviceFunctions	DeviceFunction	All device functions this end device asset performs.
[0..*]	[0..1] EndDeviceModel	EndDeviceModel	Product documentation for this end device asset.
[0..*]	[0..*] EndDeviceGroups	EndDeviceGroup	All end device groups referring to this end device asset.
[0..1]	[0..*] EndDeviceControls	EndDeviceControl	All end device controls sending commands to this end device asset.
[0..*]	[0..1] ServiceDeliveryPoint	ServiceDeliveryPoint	Service delivery point to which this end device asset belongs.
[0..*]	[0..1] ServiceLocation	ServiceLocation	Service location whose service delivery is measured by this end device asset.
[0..*]	[0..1] Customer	Customer	Customer owning this end device asset.
[0..1]	[0..*] Seals	Seal	inherited from: AssetContainer
[0..*]	[0..*] ActivityRecords	ActivityRecord	inherited from: Asset

6.2.8.10 EndDeviceControl

Instructs an EndDeviceAsset (or EndDeviceGroup) to perform a specified action.

Table 133 shows all attributes of EndDeviceControl.

Table 133 – Attributes of Metering::EndDeviceControl

name	type	description
type	String	Type of control.
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).
aliasName	String	inherited from: IdentifiedObject
description	String	inherited from: IdentifiedObject
localName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject
pathName	String	inherited from: IdentifiedObject

Table 134 shows all association ends of EndDeviceControl with other classes.

Table 134 – Association ends of Metering::EndDeviceControl with other classes

[mult from]	[mult to] name	type	description
[0..*]	[0..1] EndDeviceAsset	EndDeviceAsset	End device asset receiving commands from this end device control.
[0..*]	[0..1] EndDeviceGroup	EndDeviceGroup	End device group receiving commands from this end device control.
[0..*]	[1..1] DemandResponseProgram	DemandResponseProgram	Demand response program for this end device control.

6.2.8.11 EndDeviceEvent

Event detected by a DeviceFunction associated with EndDeviceAsset.

Table 135 shows all attributes of EndDeviceEvent.

Table 135 – Attributes of Metering::EndDeviceEvent

name	type	description
userID	String	(if user initiated) ID of user who initiated this end device event.
createdDateTime	AbsoluteDateTime	inherited from: ActivityRecord
category	String	inherited from: ActivityRecord
severity	String	inherited from: ActivityRecord
reason	String	inherited from: ActivityRecord
status	Status	inherited from: ActivityRecord

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

Table 136 shows all association ends of EndDeviceEvent with other classes.

Table 136 – Association ends of Metering::EndDeviceEvent with other classes

[mult from]	[mult to] name	type	description
[0..*]	[0..1] DeviceFunction	DeviceFunction	Device function that reported this end device event.
[0..*]	[0..1] MeterReading	MeterReading	Set of measured values to which this event applies.
[0..*]	[0..*] Assets	Asset	inherited from: ActivityRecord
[0..*]	[0..*] Documents	Document	inherited from: ActivityRecord

6.2.8.12 EndDeviceGroup

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

Table 137 shows all attributes of EndDeviceGroup.

Table 137 – Attributes of Metering::EndDeviceGroup

name	type	description
groupAddress	Integer	Address of this end device group.
aliasName	String	inherited from: IdentifiedObject
description	String	inherited from: IdentifiedObject
localName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject
pathName	String	inherited from: IdentifiedObject

Table 138 shows all association ends of EndDeviceGroup with other classes.

Table 138 – Association ends of Metering::EndDeviceGroup with other classes

[mult from]	[mult to] name	type	description
[0..*]	[0..*] EndDeviceAssets	EndDeviceAsset	All end device assets this end device group refers to.

[mult from]	[mult to] name	type	description
[0..1]	[0..*] EndDeviceControls	EndDeviceControl	All end device controls sending commands to this end device group.

6.2.8.13 IntervalBlock

Time sequence of Readings of the same ReadingType. Contained IntervalReadings may need conversion through the application of an offset and a scalar defined in associated Pending.

Table 139 shows all association ends of IntervalBlock with other classes.

Table 139 – Association ends of Metering::IntervalBlock with other classes

[mult from]	[mult to] name	type	description
[0..*]	[0..1] Pending	Pending	Pending conversion 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.

6.2.8.14 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 140 shows all attributes of IntervalReading.

Table 140 – Attributes of Metering::IntervalReading

name	type	description
value	Float	Value of this interval reading.
sensorAccuracy	PerCent	inherited from: MeasurementValue
timeStamp	AbsoluteDateTime	inherited from: MeasurementValue
aliasName	String	inherited from: IdentifiedObject
description	String	inherited from: IdentifiedObject
localName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject
pathName	String	inherited from: IdentifiedObject

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

Table 141 – 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.
[1..1]	[0..*] ReadingQualities	ReadingQuality	Used only if quality of this interval reading value is different than 'Good'.
[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

6.2.8.15 MeterAsset

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

Table 142 shows all attributes of MeterAsset.

Table 142 – Attributes of Metering::MeterAsset

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.
kR	Float	Display multiplier used to produce a displayed value from a register value.
kH	Float	Meter kh (watthour) constant. It is 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.
disconnect	Boolean	inherited from: EndDeviceAsset
loadControl	Boolean	inherited from: EndDeviceAsset
reverseFlowHandling	Boolean	inherited from: EndDeviceAsset
demandResponse	Boolean	inherited from: EndDeviceAsset
metrology	Boolean	inherited from: EndDeviceAsset
outageReport	Boolean	inherited from: EndDeviceAsset
relayCapable	Boolean	inherited from: EndDeviceAsset
readRequest	Boolean	inherited from: EndDeviceAsset
dstEnabled	Boolean	inherited from: EndDeviceAsset
timeZoneOffset	Minutes	inherited from: EndDeviceAsset
amrSystem	String	inherited from: EndDeviceAsset
phaseCount	Integer	inherited from: EndDeviceAsset
ratedCurrent	CurrentFlow	inherited from: EndDeviceAsset
ratedVoltage	Voltage	inherited from: EndDeviceAsset
category	String	inherited from: Asset
corporateCode	String	inherited from: Asset
utcNumber	String	inherited from: Asset

name	type	description
serialNumber	String	inherited from: Asset
lotNumber	String	inherited from: Asset
manufacturedDate	AbsoluteDate	inherited from: Asset
purchasePrice	Money	inherited from: Asset
critical	Boolean	inherited from: Asset
application	String	inherited from: Asset
installationDate	AbsoluteDate	inherited from: Asset
acceptanceTest	AcceptanceTest	inherited from: Asset
initialCondition	String	inherited from: Asset
initialLossOfLife	PerCent	inherited from: Asset
status	Status	inherited from: Asset
electronicAddress	ElectronicAddress	inherited from: Asset
aliasName	String	inherited from: IdentifiedObject
description	String	inherited from: IdentifiedObject
localName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject
pathName	String	inherited from: IdentifiedObject

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

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

[mult from]	[mult to] name	type	description
[0..1]	[0..*] MeterReadings	MeterReading	All meter readings provided by this meter asset.
[0..1]	[0..*] MeterServiceWorks	MeterServiceWork	All non-replacement works on this meter asset.
[0..1]	[0..*] VendingTransactions	Transaction	All vending transactions on this meter asset.
[0..1]	[0..*] MeterReplacementWorks	MeterServiceWork	All works on replacement of this old meter asset.
[0..1]	[0..*] DeviceFunctions	DeviceFunction	inherited from: EndDeviceAsset
[0..*]	[0..1] EndDeviceModel	EndDeviceModel	inherited from: EndDeviceAsset
[0..*]	[0..*] EndDeviceGroups	EndDeviceGroup	inherited from: EndDeviceAsset
[0..1]	[0..*] EndDeviceControls	EndDeviceControl	inherited from: EndDeviceAsset
[0..*]	[0..1] ServiceDeliveryPoint	ServiceDeliveryPoint	inherited from: EndDeviceAsset
[0..*]	[0..1] ServiceLocation	ServiceLocation	inherited from: EndDeviceAsset
[0..*]	[0..1] Customer	Customer	inherited from: EndDeviceAsset
[0..1]	[0..*] Seals	Seal	inherited from: AssetContainer
[0..*]	[0..*] ActivityRecords	ActivityRecord	inherited from: Asset

6.2.8.16 MeterReading

Set of values obtained from the meter.

Table 144 shows all attributes of MeterReading.

Table 144 – Attributes of Metering::MeterReading

name	type	description
valuesInterval	DateTimeInterval	Date and time interval of the data items contained within this meter reading.
aliasName	String	inherited from: IdentifiedObject
description	String	inherited from: IdentifiedObject
localName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject
pathName	String	inherited from: IdentifiedObject

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

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

[mult from]	[mult to] name	type	description
[0..*]	[0..1] ServiceDeliveryPoint	ServiceDeliveryPoint	Service delivery point from which this meter reading (set of values) has been obtained.
[0..1]	[0..*] EndDeviceEvents	EndDeviceEvent	All end device events associated with this set of measured values.
[0..*]	[0..1] CustomerAgreement	CustomerAgreement	(could be deprecated in the future) Customer agreement for this meter reading.
[0..*]	[0..1] MeterAsset	MeterAsset	Meter asset providing this meter reading.
[0..*]	[0..*] Readings	Reading	All reading values contained within this meter reading.
[0..1]	[0..*] IntervalBlocks	IntervalBlock	All interval blocks contained in this meter reading.

6.2.8.17 MeterServiceWork

Work involving meters.

Table 146 shows all attributes of MeterServiceWork.

Table 146 – Attributes of Metering::MeterServiceWork

name	type	description
kind	WorkKind	inherited from: Work
priority	String	inherited from: Work
requestDateTime	AbsoluteDateTime	inherited from: Work
category	String	inherited from: Document
createdDateTime	AbsoluteDateTime	inherited from: Document
lastModifiedDateTime	AbsoluteDateTime	inherited from: Document
revisionNumber	String	inherited from: Document

name	type	description
subject	String	inherited from: Document
title	String	inherited from: Document
docStatus	Status	inherited from: Document
status	Status	inherited from: Document
electronicAddress	ElectronicAddress	inherited from: Document
aliasName	String	inherited from: IdentifiedObject
description	String	inherited from: IdentifiedObject
localName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject
pathName	String	inherited from: IdentifiedObject

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

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

[mult from]	[mult to] name	type	description
[0..*]	[0..1] MeterAsset	MeterAsset	Meter asset on which this non-replacement work is performed.
[0..*]	[0..1] OldMeterAsset	MeterAsset	Old meter asset replaced by this work.
[0..*]	[0..*] Customers	Customer	inherited from: Work
[0..*]	[0..*] ActivityRecords	ActivityRecord	inherited from: Document

6.2.8.18 Pending

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

Table 148 shows all attributes of Pending.

Table 148 – Attributes of Metering::Pending

name	type	description
scalarFloat	Float	(if scalar is floating number) When multiplied with 'IntervalReading.value', it causes a unit of measure conversion to occur, resulting in 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 this attribute and divided by 'scalarDenominator, it causes a unit of measure conversion to occur,

name	type	description
		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 149 shows all association ends of Pending with other classes.

Table 149 – Association ends of Metering::Pending 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.

6.2.8.19 Reading

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

Table 150 shows all attributes of Reading.

Table 150 – Attributes of Metering::Reading

name	type	description
value	Float	Value of this reading.
sensorAccuracy	PerCent	inherited from: MeasurementValue
timeStamp	AbsoluteDateTime	inherited from: MeasurementValue
aliasName	String	inherited from: IdentifiedObject
description	String	inherited from: IdentifiedObject
localName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject
pathName	String	inherited from: IdentifiedObject

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

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

[mult from]	[mult to] name	type	description
[0..*]	[1..1] ReadingType	ReadingType	Type information for this reading value.
[0..*]	[0..*] MeterReadings	MeterReading	All meter readings (sets of values) containing this reading value.
[0..1]	[0..*] ReadingQualities	ReadingQuality	Used only if quality of this reading value is different than 'Good'.
[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

6.2.8.20 ReadingQuality

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 ReadingQuality).

Table 152 shows all attributes of ReadingQuality.

Table 152 – Attributes of Metering::ReadingQuality

name	type	description
quality	String	Quality, to be specified if different than 'Good'.

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

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

[mult from]	[mult to] name	type	description
[0..*]	[0..1] Reading	Reading	Reading value to which this quality applies.
[0..*]	[1..1] IntervalReading	IntervalReading	Interval reading value to which this quality applies.

6.2.8.21 ReadingType

Type of data conveyed by a specific Reading.

Table 154 shows all attributes of ReadingType.

Table 154 – Attributes of Metering::ReadingType

name	type	description
kind	ReadingKind	Kind of reading.
unit	UnitSymbol	Unit for the reading value.
multiplier	UnitMultiplier	Multiplier for 'unit'.
intervalLength	Seconds	(if incremental reading value) Length of increment interval.
reverseChronology	Boolean	True for systems that must operate in "reverse" chronological order.
defaultValueDataType	String	Numeric type to be expected for the associated IntervalBlock.value (e.g. unsignedInteger).
defaultQuality	String	Characteristics of a data value conveyed by a specific Reading, which allow an application to understand how a specific Reading is to be interpreted.

name	type	description
dynamicConfiguration	DynamicDemand	Demand configuration.
channelNumber	Integer	Logical positioning of this measurement data.
aliasName	String	inherited from: IdentifiedObject
description	String	inherited from: IdentifiedObject
localName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject
pathName	String	inherited from: IdentifiedObject

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

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

[mult from]	[mult to] name	type	description
[0..1]	[0..1] Register	Register	Register displaying values with this type information.
[1..1]	[0..*] Readings	Reading	All reading values with this type information.
[1..1]	[0..*] IntervalBlocks	IntervalBlock	All blocks containing interval reading values with this type information.
[1..1]	[0..1] Pending	Pending	Pending conversion that produced this reading type.

6.2.8.22 Register

Display for quantity that is metered on an end device such as a meter.

Table 156 shows all attributes of Register.

Table 156 – Attributes of Metering::Register

name	type	description
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 5.
aliasName	String	inherited from: IdentifiedObject
description	String	inherited from: IdentifiedObject
localName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject
pathName	String	inherited from: IdentifiedObject

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

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

[mult from]	[mult to] name	type	description
[0..*]	[0..1] DeviceFunction	DeviceFunction	Device function metering quantities displayed by this register.
[0..1]	[0..1] ReadingType	ReadingType	Reading type for values displayed by this register.

6.2.8.23 SDPLocation

Location of an individual service delivery point. For residential or most businesses, it is typically the location of a meter on the customer's premises. For transmission, it is the point(s) of interconnection on the transmission provider's transmission system where capacity and/or energy transmitted by the transmission provider is made available to the receiving party. The point(s) of delivery is specified in the service agreement.

Table 158 shows all attributes of SDPLocation.

Table 158 – Attributes of Metering::SDPLocation

name	type	description
occupancyDate	AbsoluteDate	Date when certificate of occupancy was provided for this location, 0 if valid certificate of occupancy does not exist for (inherited) 'Location.corporateCode'.
accessMethod	String	Method for the service person to access this service delivery 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 service delivery point location. Examples include: bad dog, violent customer, verbally abusive occupant, obstructions, safety hazards, etc.
remark	String	Remarks about this location.
category	String	inherited from: Location
corporateCode	String	inherited from: Location
mainAddress	StreetAddress	inherited from: Location
secondaryAddress	StreetAddress	inherited from: Location
direction	String	inherited from: Location
geoInfoReference	String	inherited from: Location
status	Status	inherited from: Location
phone1	TelephoneNumber	inherited from: Location
phone2	TelephoneNumber	inherited from: Location
electronicAddress	ElectronicAddress	inherited from: Location
aliasName	String	inherited from: IdentifiedObject
description	String	inherited from: IdentifiedObject
localName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject
pathName	String	inherited from: IdentifiedObject

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

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

[mult from]	[mult to] name	type	description
[0..*]	[0..*] ServiceDeliveryPoints	ServiceDeliveryPoint	All service delivery points at this location.
[0..1]	[0..*] PowerSystemResources	PowerSystemResource	inherited from: Location
[1..1]	[0..*] CoordinateSystems	CoordinateSystem	inherited from: Location

6.2.8.24 ServiceDeliveryPoint

Logical point on the network where the ownership of the service changes hands. It is one of potentially many service points within a ServiceLocation, delivering service in accordance with a CustomerAgreement. Used at the place where a meter may be installed.

Table 160 shows all attributes of ServiceDeliveryPoint.

Table 160 – Attributes of Metering::ServiceDeliveryPoint

name	type	description
phaseCode	PhaseCode	Phase code. Number of wires and number of phases can be deduced from enumeration literal values. For example, ABCN is three-phase, four-wire. s12n (splitSecondary12N) is single-phase, three-wire. s1n and s2n are single-phase, two-wire.
ctptReference	Integer	(optional for medium voltage connections) Reference to the low side terminal of a CT or PT that obtain readings from a medium or high voltage point.
grounded	Boolean	True if grounded.
nominalServiceVoltage	Integer	Nominal service voltage.
servicePriority	String	Priority of service for this service delivery point. Note that service delivery points at the same service location can have different priorities.
serviceDeliveryRemark	String	Remarks about this service delivery 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 that this service delivery point is configured to deliver.
ratedPower	ActivePower	Power that this service delivery point is configured to deliver.
aliasName	String	inherited from: IdentifiedObject
description	String	inherited from: IdentifiedObject
localName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject
pathName	String	inherited from: IdentifiedObject

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

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

[mult from]	[mult to] name	type	description
[0..1]	[0..*] EndDeviceAssets	EndDeviceAsset	All end device assets at this service delivery point.
[0..*]	[0..1] ServiceCategory	ServiceCategory	Service category delivered by this service delivery point.
[0..*]	[0..1] ServiceSupplier	ServiceSupplier	ServiceSupplier (Utility) utilising this service delivery point to deliver a service.
[0..*]	[0..*] SDPLocations	SDPLocation	All locations of this service delivery point.
[0..1]	[0..*] MeterReadings	MeterReading	All meter readings obtained from this service delivery point.
[0..*]	[0..1] ServiceLocation	ServiceLocation	Service location where the service delivered by this service delivery point is consumed.
[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] CustomerAgreement	CustomerAgreement	Customer agreement regulating this service delivery point.

6.2.9 Package LoadControl

6.2.9.1 General

This package is an extension of the Metering package and contains the information classes that support specialized 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 47 shows logical diagram LoadControlInheritance.

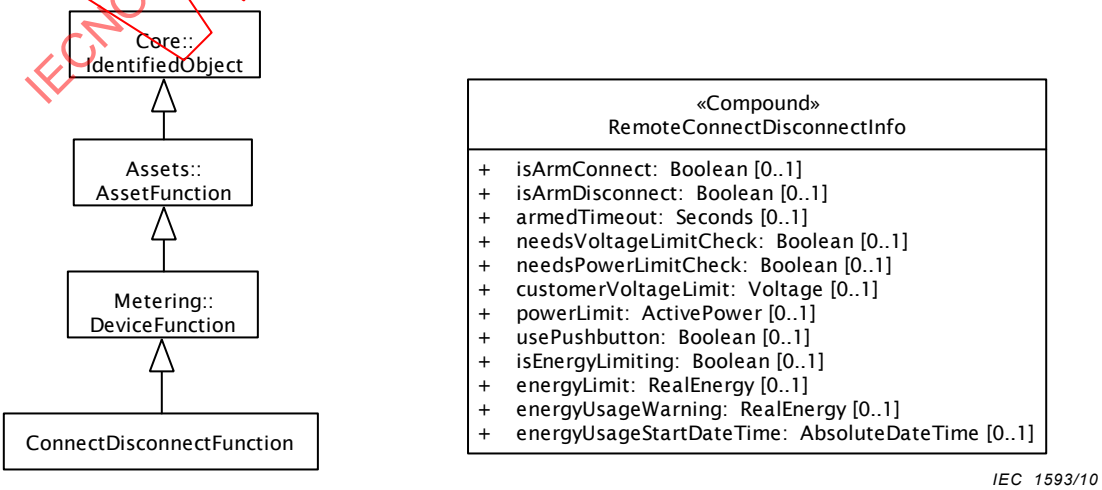


Figure 47 – Logical diagram LoadControl::LoadControlInheritance

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

Figure 48 shows logical diagram LoadControlOverview.

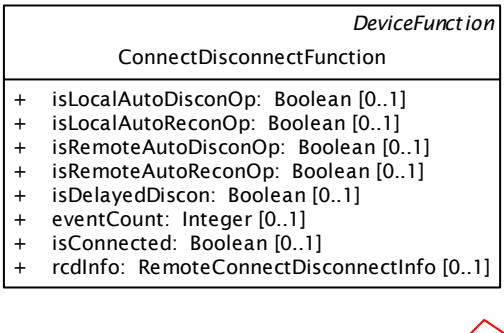


Figure 48 – Logical diagram LoadControl::LoadControlOverview

This diagram shows normative classes from this package.

6.2.9.2 RemoteConnectDisconnectInfo compound

Details of remote connect disconnect function.

Table 162 shows all attributes of RemoteConnectDisconnectInfo.

Table 162 – Attributes of LoadControl::RemoteConnectDisconnectInfo

name	type	description
isArmConnect	Boolean	True if the RCD switch must be armed before a connect action can be initiated.
isArmDisconnect	Boolean	True if the RCD switch must be armed before a disconnect action can be initiated.
armedTimeout	Seconds	Setting of the timeout elapsed time.
needsVoltageLimitCheck	Boolean	True if voltage limit must be checked to prevent connect if voltage is over the limit.
needsPowerLimitCheck	Boolean	True if load limit must 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 must 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	AbsoluteDateTime	Start date and time to accumulate energy for energy usage limiting.

6.2.9.3 ConnectDisconnectFunction

A function that will disconnect or reconnect the customer's load under defined conditions.

Table 163 shows all attributes of ConnectDisconnectFunction.

Table 163 – Attributes of LoadControl::ConnectDisconnectFunction

name	type	description
isLocalAutoDisconOp	Boolean	(if disconnection can be operated locally) If set true, 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 is 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 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 had been given. If set false, the switch may disconnect the service immediately after the disconnect signal had 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.
disabled	Boolean	inherited from: DeviceFunction
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
description	String	inherited from: IdentifiedObject
localName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject
pathName	String	inherited from: IdentifiedObject

Table 164 shows all association ends of ConnectDisconnectFunction with other classes.

Table 164 – Association ends of LoadControl::ConnectDisconnectFunction with other classes

[mult from]	[mult to] name	type	description
[0..1]	[0..*] Registers	Register	inherited from: DeviceFunction
[0..*]	[0..1] EndDeviceAsset	EndDeviceAsset	inherited from: DeviceFunction
[0..1]	[0..*] EndDeviceEvents	EndDeviceEvent	inherited from: DeviceFunction

6.2.10 Package PaymentMetering

6.2.10.1 General

This package is an extension of the Metering package and contains the information classes that support specialized 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 49 shows logical diagram PaymentMeteringInheritance.

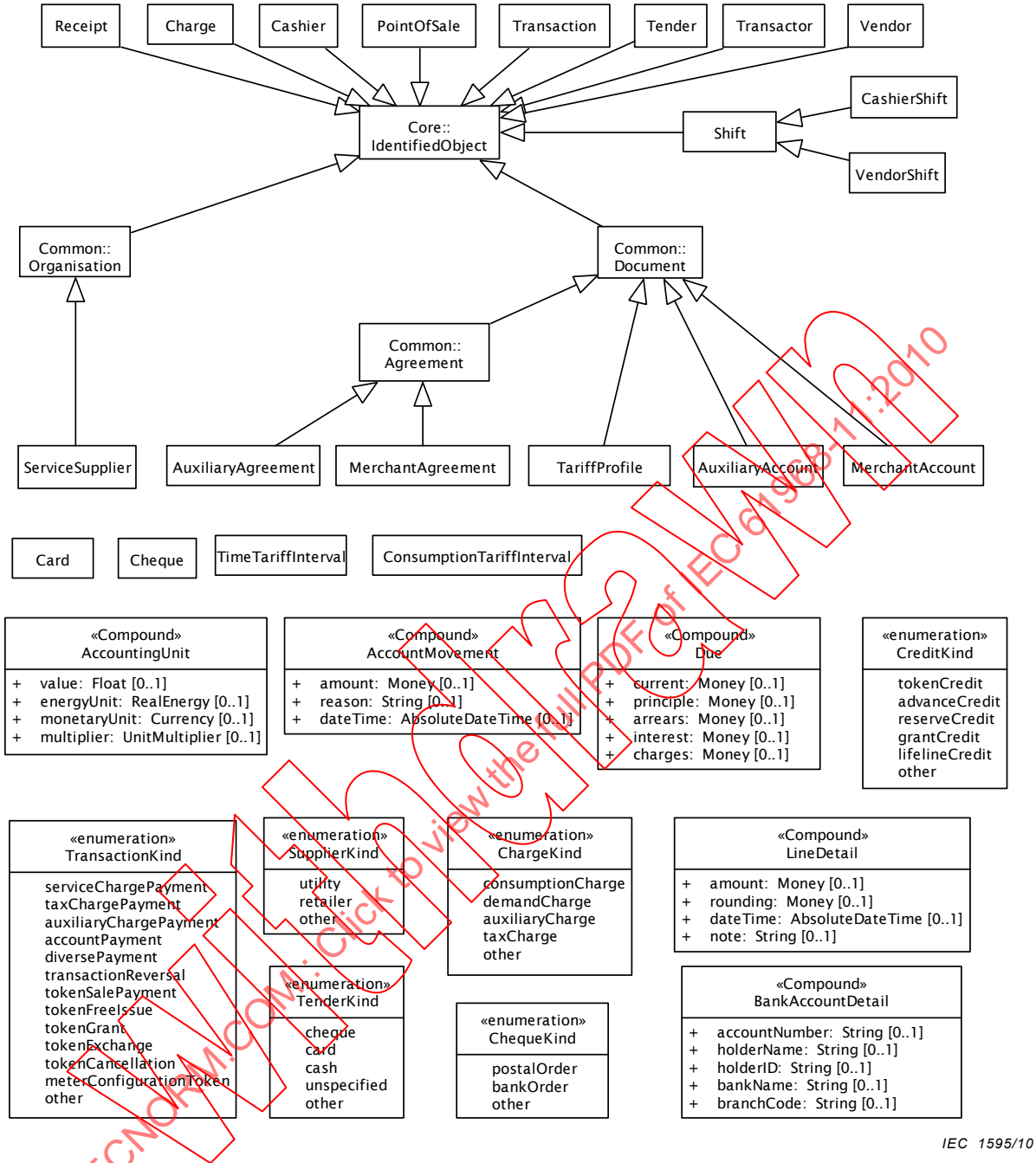


Figure 49 – Logical diagram PaymentMetering::PaymentMeteringInheritance

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

Figure 50 shows logical diagram PaymentMeteringOverview.

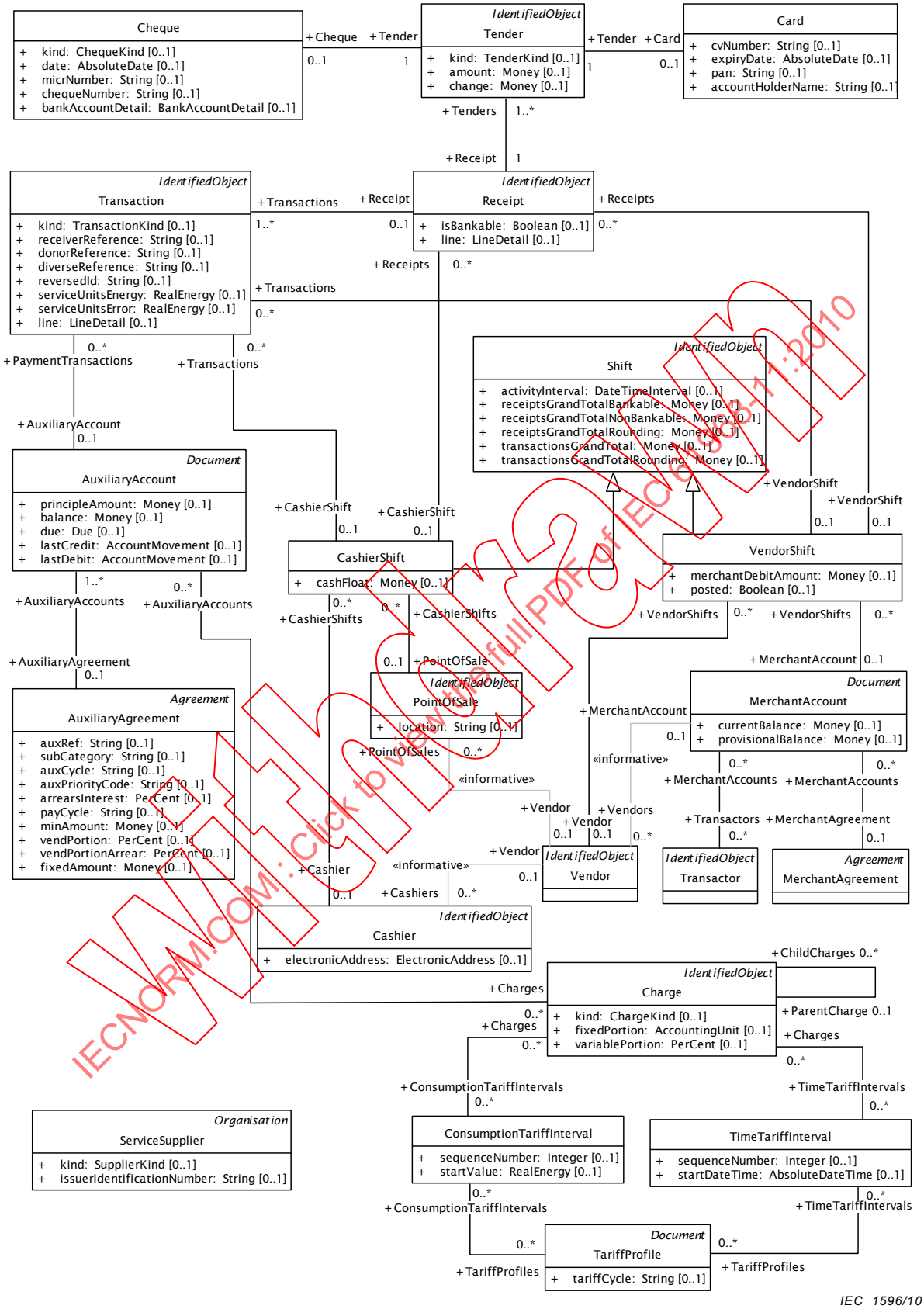


Figure 50 – Logical diagram PaymentMetering::PaymentMeteringOverview

This diagram shows normative classes from this package.

Figure 51 shows logical diagram PaymentMeteringRelationships.

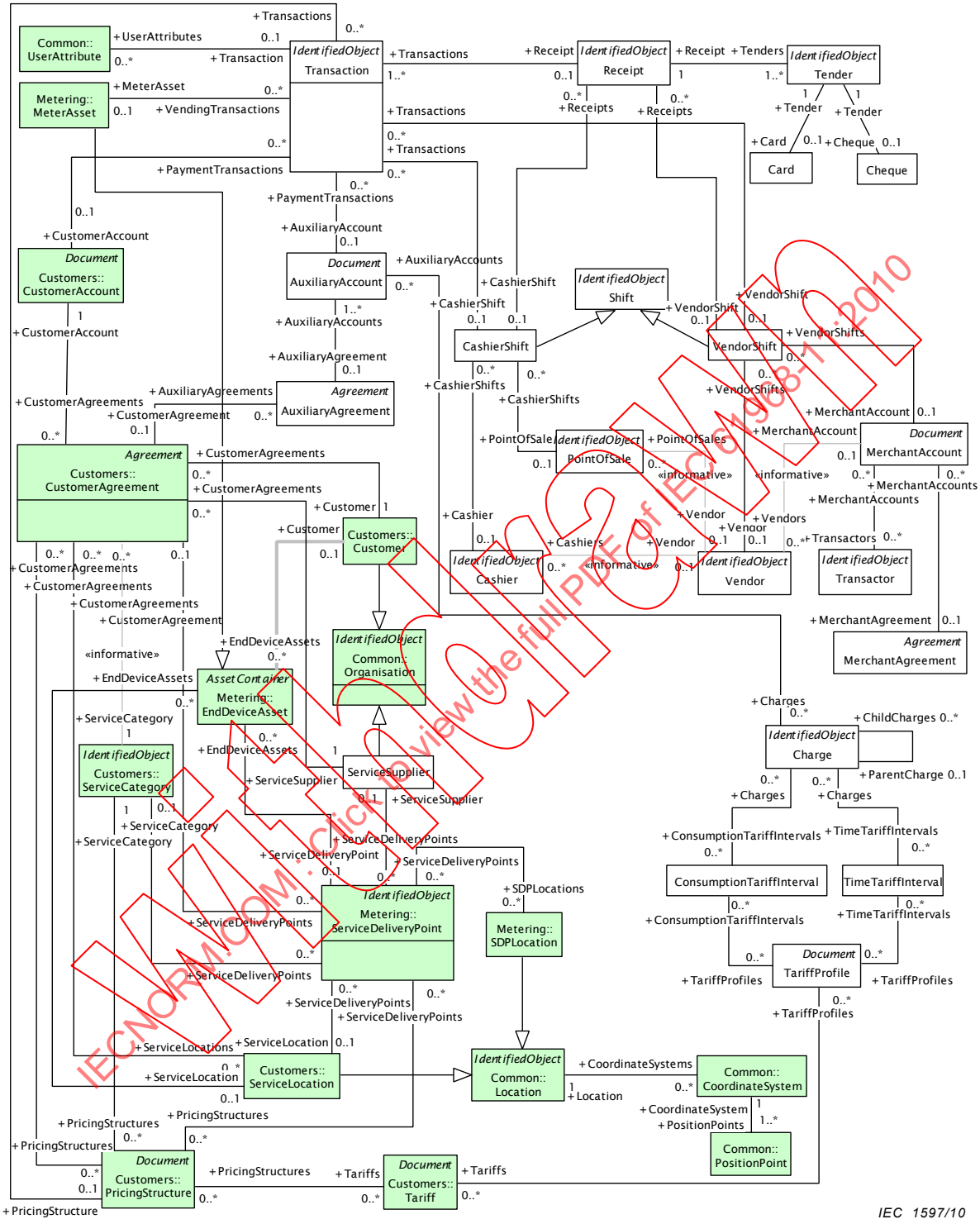


Figure 51 – Logical diagram PaymentMetering::PaymentMeteringRelationships

This diagram shows relationships of the classes from this package with those from other packages.

Figure 52 shows logical diagram Transacting.

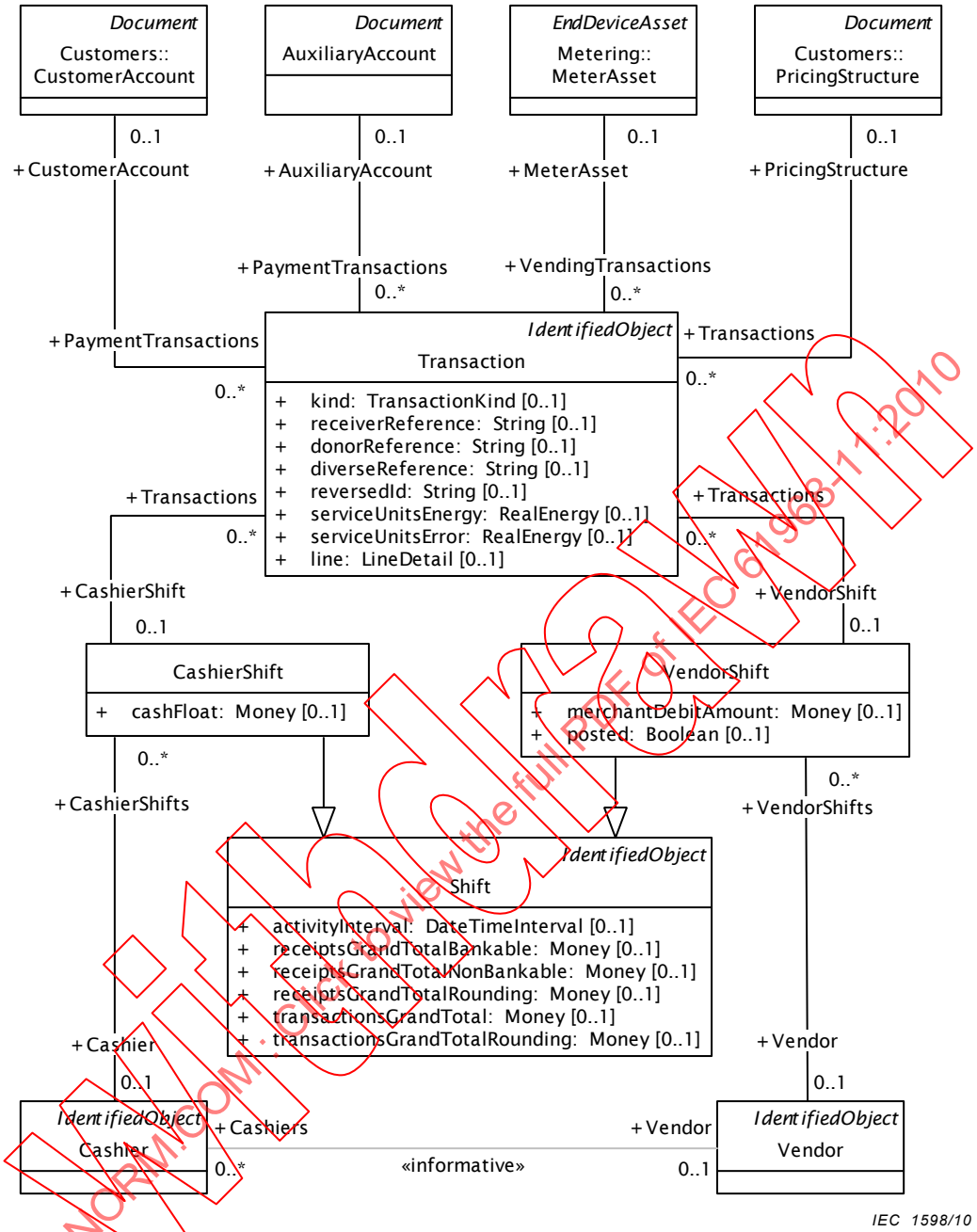


Figure 52 – Logical diagram PaymentMetering::Transacting

This diagram shows classes used to model transacting.

Figure 53 shows logical diagram Receipting.

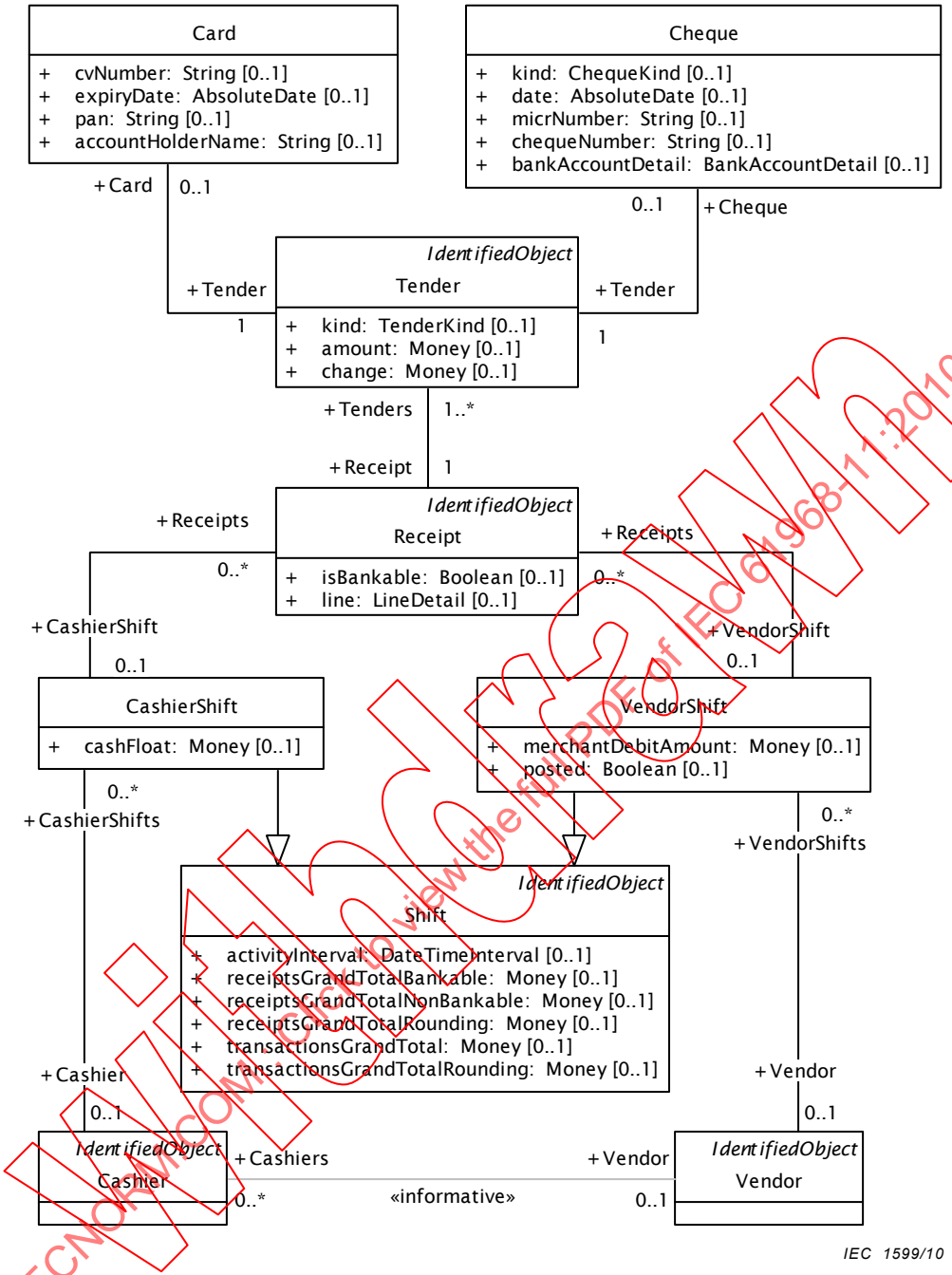


Figure 53 – Logical diagram PaymentMetering::Receipting

This diagram shows classes used to model receipting.

Figure 54 shows logical diagram AuxiliaryAgreement.

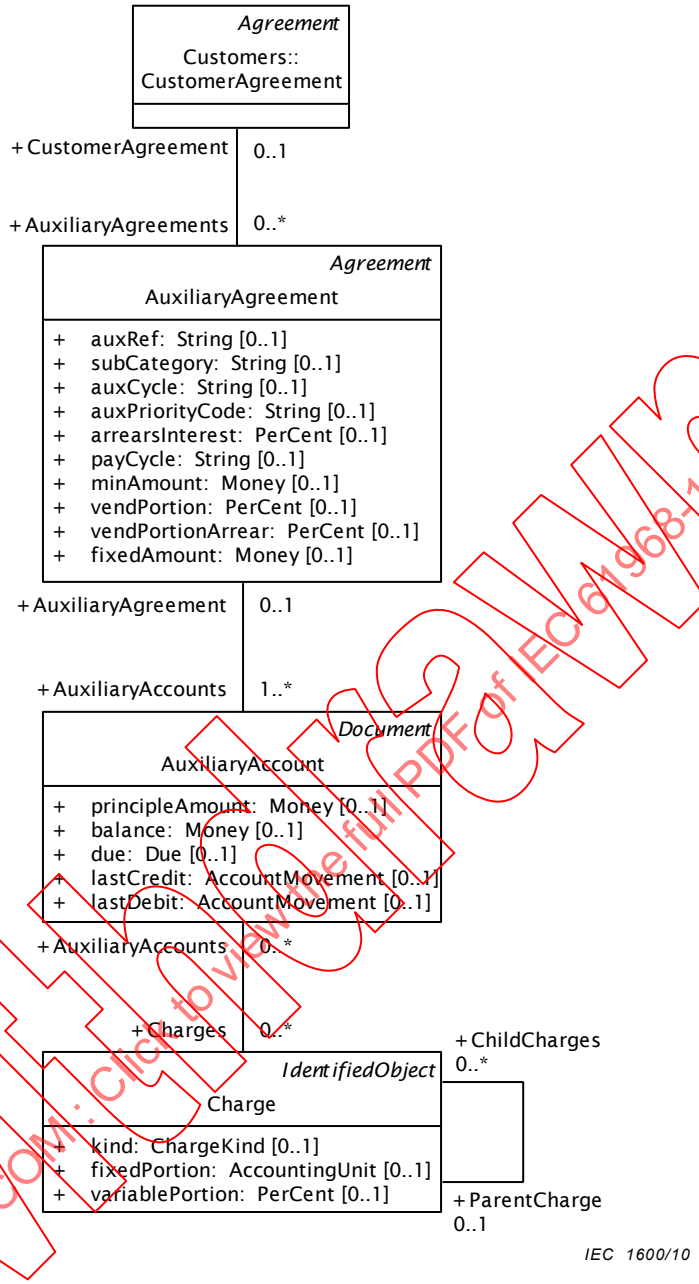


Figure 54 – Logical diagram PaymentMetering::AuxiliaryAgreement

This diagram shows classes used to model auxiliary agreements.

Figure 55 shows logical diagram TariffProfile.

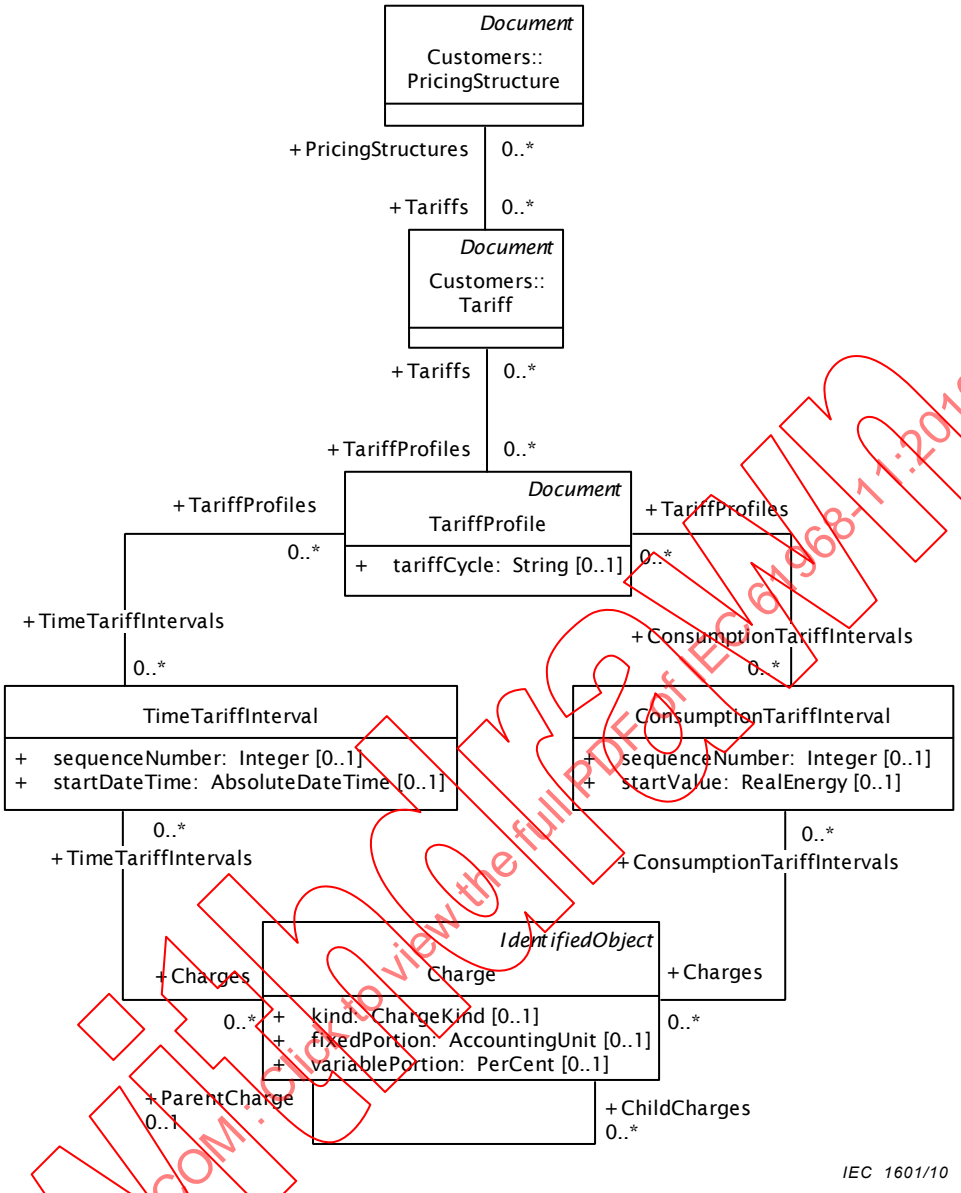


Figure 55 – Logical diagram PaymentMetering::TariffProfile

This diagram shows classes used to model tariff profiles.

6.2.10.2 AccountMovement compound

Credit/debit movements for an account.

Table 165 shows all attributes of AccountMovement.

Table 165 – 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	AbsoluteDateTime	Date and time when the credit/debit transaction

name	type	description
		was performed.

6.2.10.3 AccountingUnit compound

Unit for accounting; use either 'energyUnit' or 'currencyUnit' to specify the unit for 'value'.

Table 166 shows all attributes of AccountingUnit.

Table 166 – 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'.

6.2.10.4 BankAccountDetail compound

Details of a bank account.

Table 167 shows all attributes of BankAccountDetail.

Table 167 – 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.

6.2.10.5 Due compound

Details on amounts due for an account.

Table 168 shows all attributes of Due.

Table 168 – 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.

name	type	description
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'.

6.2.10.6 LineDetail compound

Details on an amount line, with rounding, date and note.

Table 169 shows all attributes of LineDetail.

Table 169 – 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	AbsoluteDateTime	Date and time when this line was created in the application process.
note	String	Free format note relevant to this line.

6.2.10.7 ChargeKind enumeration

Kind of charge.

Table 170 shows all literals of ChargeKind.

Table 170 – 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	

6.2.10.8 ChequeKind enumeration

Kind of cheque.

Table 171 shows all literals of ChequeKind.

Table 171 – Literals of PaymentMetering::ChequeKind

literal	description
postalOrder	
bankOrder	
other	

6.2.10.9 CreditKind enumeration

Kind of credit.

Table 172 shows all literals of CreditKind.

Table 172 – Literals of PaymentMetering::CreditKind

literal	description
tokenCredit	
advanceCredit	
reserveCredit	
grantCredit	
lifelineCredit	
other	

6.2.10.10 SupplierKind enumeration

Kind of supplier.

Table 173 shows all literals of SupplierKind.

Table 173 – Literals of PaymentMetering::SupplierKind

literal	description
utility	
retailer	
other	

6.2.10.11 TenderKind enumeration

Kind of tender.

Table 174 shows all literals of TenderKind.

Table 174 – Literals of PaymentMetering::TenderKind

literal	description
cheque	
card	
cash	

literal	description
unspecified	
other	

6.2.10.12 TransactionKind enumeration

Kind of transaction.

Table 175 shows all literals of TransactionKind.

Table 175 – Literals of PaymentMetering::TransactionKind

literal	description
serviceChargePayment	
taxChargePayment	
auxiliaryChargePayment	
accountPayment	
diversePayment	
transactionReversal	
tokenSalePayment	
tokenFreeIssue	
tokenGrant	
tokenExchange	
tokenCancellation	
meterConfigurationToken	
other	

6.2.10.13 AuxiliaryAccount

Variable and dynamic part of AuxiliaryAgreement, generally representing the current state of the account related to the outstanding balance defined in AuxiliaryAgreement.

Table 176 shows all attributes of AuxiliaryAccount.

Table 176 – 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.
category	String	inherited from: Document

name	type	description
createdDateTime	AbsoluteDateTime	inherited from: Document
lastModifiedDateTime	AbsoluteDateTime	inherited from: Document
revisionNumber	String	inherited from: Document
subject	String	inherited from: Document
title	String	inherited from: Document
docStatus	Status	inherited from: Document
status	Status	inherited from: Document
electronicAddress	ElectronicAddress	inherited from: Document
aliasName	String	inherited from: IdentifiedObject
description	String	inherited from: IdentifiedObject
localName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject
pathName	String	inherited from: IdentifiedObject

Table 177 shows all association ends of AuxiliaryAccount with other classes.

Table 177 – 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..*]	[0..*] ActivityRecords	ActivityRecord	inherited from: Document

6.2.10.14 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 owing 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 he needs to purchase a prepaid token for electricity.

The present status of AuxiliaryAgreement 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 178 shows all attributes of AuxiliaryAgreement.

Table 178 – Attributes of PaymentMetering::AuxiliaryAgreement

name	type	description
auxRef	String	A local reference to this AuxiliaryAgreement defined in the context of the implementation and not related to IdentifiedObject.mRID.
subCategory	String	Sub-category of this AuxiliaryAgreement as sub-

name	type	description
		classification of the inherited 'category'.
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 AuxiliaryAgreement has above other AuxiliaryAgreements (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 must be paid at any transaction towards settling this AuxiliryAgreement or reducing the balance.
vendPortion	PerCent	The percentage of the transaction amount that must be collected from each vending transaction towards settlement of this AuxiliaryAgreement 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 must be collected from each vending transaction towards settlement of this AuxiliaryAgreement 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 must be collected from each vending transaction towards settlement of this AuxiliaryAgreement. Note that there may be multiple tokens vended per vending transaction, but this is not relevant.
signDate	AbsoluteDate	inherited from: Agreement
validityInterval	DateTimeInterval	inherited from: Agreement
category	String	inherited from: Document
createdDateTime	AbsoluteDateTime	inherited from: Document
lastModifiedDateTime	AbsoluteDateTime	inherited from: Document
revisionNumber	String	inherited from: Document
subject	String	inherited from: Document
title	String	inherited from: Document
docStatus	Status	inherited from: Document
status	Status	inherited from: Document
electronicAddress	ElectronicAddress	inherited from: Document
aliasName	String	inherited from: IdentifiedObject
description	String	inherited from: IdentifiedObject
localName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject
pathName	String	inherited from: IdentifiedObject

Table 179 shows all association ends of AuxiliaryAgreement with other classes.

Table 179 – 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..*]	[0..*] ActivityRecords	ActivityRecord	inherited from: Document

6.2.10.15 Card

Documentation of the tender when it is a type of card (credit, debit, etc).

Table 180 shows all attributes of Card.

Table 180 – Attributes of PaymentMetering::Card

name	type	description
cvNumber	String	The card verification number.
expiryDate	AbsoluteDate	The date when this card expires.
pan	String	The primary account number.
accountHolderName	String	Name of account holder.

Table 181 shows all association ends of Card with other classes.

Table 181 – 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.

6.2.10.16 Cashier

The operator of the point of sale for the duration of CashierShift. Cashier is under the exclusive management control of Vendor.

Table 182 shows all attributes of Cashier.

Table 182 – Attributes of PaymentMetering::Cashier

name	type	description
electronicAddress	ElectronicAddress	Electronic address.
aliasName	String	inherited from: IdentifiedObject
description	String	inherited from: IdentifiedObject
localName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject
pathName	String	inherited from: IdentifiedObject

Table 183 shows all association ends of Cashier with other classes.

Table 183 – 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.

6.2.10.17 CashierShift

The operating shift for a cashier, during which he may transact against the CashierShift, subject to VendorShift being open.

Table 184 shows all attributes of CashierShift.

Table 184 – Attributes of PaymentMetering::CashierShift

name	type	description
cashFloat	Money	The amount of cash that the cashier brings with him to start his shift and that he will take away at the end of his 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
description	String	inherited from: IdentifiedObject
localName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject
pathName	String	inherited from: IdentifiedObject

Table 185 shows all association ends of CashierShift with other classes.

Table 185 – 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	
[0..*]	[0..1] Cashier	Cashier	Cashier operating this shift.

6.2.10.18 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 fixedPortion plus percentagePortion.

Table 186 shows all attributes of Charge.

Table 186 – 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
description	String	inherited from: IdentifiedObject
localName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject
pathName	String	inherited from: IdentifiedObject

Table 187 shows all association ends of Charge with other classes.

Table 187 – 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 must be levied.
[0..*]	[0..*] ConsumptionTariffIntervals	ConsumptionTariffInterval	Tariff intervals to which this consumption-based charge must be levied.
[0..*]	[0..*] TimeTariffIntervals	TimeTariffInterval	Tariff intervals to which this time-based charge must be levied.
[0..*]	[0..1] ParentCharge	Charge	Parent of this charge.

6.2.10.19 Cheque

The actual tender when it is a type of cheque.

Table 188 shows all attributes of Cheque.

Table 188 – Attributes of PaymentMetering::Cheque

name	type	description
kind	ChequeKind	Kind of cheque.
date	AbsoluteDate	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 189 shows all association ends of Cheque with other classes.

Table 189 – 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.

6.2.10.20 ConsumptionTariffInterval

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 190 shows all attributes of ConsumptionTariffInterval.

Table 190 – 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 191 shows all association ends of ConsumptionTariffInterval with other classes.

Table 191 – 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..*] TariffProfiles	TariffProfile	All tariff profiles defined by this consumption tariff interval.

6.2.10.21 MerchantAccount

The operating account controlled by MerchantAgreement, against which Vendor may vend tokens or receipt payments. Transactions via VendorShift debit the account and bank deposits via BankStatement credit the account.

Table 192 shows all attributes of MerchantAccount.

Table 192 – 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).
category	String	inherited from: Document
createdDateTime	AbsoluteDateTime	inherited from: Document
lastModifiedDateTime	AbsoluteDateTime	inherited from: Document
revisionNumber	String	inherited from: Document
subject	String	inherited from: Document
title	String	inherited from: Document
docStatus	Status	inherited from: Document
status	Status	inherited from: Document
electronicAddress	ElectronicAddress	inherited from: Document
aliasName	String	inherited from: IdentifiedObject
description	String	inherited from: IdentifiedObject
localName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject
pathName	String	inherited from: IdentifiedObject

Table 193 shows all association ends of MerchantAccount with other classes.

Table 193 – 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.
[0..*]	[0..*] ActivityRecords	ActivityRecord	inherited from: Document

6.2.10.22 MerchantAgreement

A formal controlling contractual agreement between supplier and merchant, in terms of which merchant is authorised to vend tokens and receipt payments on behalf of supplier. Merchant is accountable to supplier for revenue collected at PointOfSale.

Table 194 shows all attributes of MerchantAgreement.

Table 194 – Attributes of PaymentMetering::MerchantAgreement

name	type	description
signDate	AbsoluteDate	inherited from: Agreement
validityInterval	DateTimeInterval	inherited from: Agreement
category	String	inherited from: Document
createdDateTime	AbsoluteDateTime	inherited from: Document
lastModifiedDateTime	AbsoluteDateTime	inherited from: Document
revisionNumber	String	inherited from: Document
subject	String	inherited from: Document
title	String	inherited from: Document
docStatus	Status	inherited from: Document
status	Status	inherited from: Document
electronicAddress	ElectronicAddress	inherited from: Document
aliasName	String	inherited from: IdentifiedObject
description	String	inherited from: IdentifiedObject
localName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject
pathName	String	inherited from: IdentifiedObject

Table 195 shows all association ends of MerchantAgreement with other classes.

Table 195 – 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..*]	[0..*] ActivityRecords	ActivityRecord	inherited from: Document

6.2.10.23 PointOfSale

Logical point where transactions take place with operational interaction between cashier and the payment system; in certain cases PointOfSale interacts directly with the end customer, in which case cashier might not be a real person: for example a self-service kiosk or over the internet.

Table 196 shows all attributes of PointOfSale.

Table 196 – 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
description	String	inherited from: IdentifiedObject
localName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject
pathName	String	inherited from: IdentifiedObject

Table 197 shows all association ends of PointOfSale with other classes.

Table 197 – 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.

6.2.10.24 Receipt

Record of total receipted payment from customer.

Table 198 shows all attributes of Receipt.

Table 198 – 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
description	String	inherited from: IdentifiedObject
localName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject
pathName	String	inherited from: IdentifiedObject

Table 199 shows all association ends of Receipt with other classes.

Table 199 – 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.
[1..1]	[1..*] Tenders	Tender	All payments received in the form of tenders recorded by this receipt.

6.2.10.25 ServiceSupplier

Organisation that provides services to customers.

Table 200 shows all attributes of ServiceSupplier.

Table 200 – 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 Standards Organisation for the purpose of tagging onto electronic financial transactions, as defined in ISO/IEC 7812-1 and ISO/IEC 7812-2.
streetAddress	StreetAddress	inherited from: Organisation
postalAddress	PostalAddress	inherited from: Organisation
phone1	TelephoneNumber	inherited from: Organisation
phone2	TelephoneNumber	inherited from: Organisation
electronicAddress	ElectronicAddress	inherited from: Organisation

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

Table 201 shows all association ends of ServiceSupplier with other classes.

Table 201 – Association ends of PaymentMetering::ServiceSupplier with other classes

[mult from]	[mult to] name	type	description
[0..1]	[0..*] ServiceDeliveryPoints	ServiceDeliveryPoint	All service delivery points this service supplier utilises to deliver a service.
[1..1]	[0..*] CustomerAgreements	CustomerAgreement	All customer agreements of this service supplier.

6.2.10.26 Shift

Generally referring to a period of operation or work performed. Whether shift is open/closed can be derived from attributes 'activiryInterval.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 attributes:

=sum(Receipt.receiptAmount) ; includes bankable and non-bankable receipts.

Must also reconcile against:

=sum(receiptsGrandTotalBankable + receiptsGrandTotalNonBankable).

Must also reconcile 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 202 shows all attributes of Shift.

Table 202 – 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 must also reconcile 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
description	String	inherited from: IdentifiedObject
localName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject
pathName	String	inherited from: IdentifiedObject

6.2.10.27 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 203 shows all attributes of TariffProfile.

Table 203 – 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.
category	String	inherited from: Document
createdDateTime	AbsoluteDateTime	inherited from: Document
lastModifiedDateTime	AbsoluteDateTime	inherited from: Document
revisionNumber	String	inherited from: Document
subject	String	inherited from: Document
title	String	inherited from: Document
docStatus	Status	inherited from: Document
status	Status	inherited from: Document
electronicAddress	ElectronicAddress	inherited from: Document
aliasName	String	inherited from: IdentifiedObject
description	String	inherited from: IdentifiedObject
localName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject
pathName	String	inherited from: IdentifiedObject

Table 204 shows all association ends of TariffProfile with other classes.

Table 204 – 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..*]	[0..*] ActivityRecords	ActivityRecord	inherited from: Document

6.2.10.28 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' must exist separately in each of Cheque and Card as each could have a different account holder name.

Table 205 shows all attributes of Tender.

Table 205 – 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
description	String	inherited from: IdentifiedObject
localName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject
pathName	String	inherited from: IdentifiedObject

Table 206 shows all association ends of Tender with other classes.

Table 206 – 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] Cheque	Cheque	Cheque used to tender payment.
[1..1]	[0..1] Card	Card	Card used to tender payment.

6.2.10.29 TimeTariffInterval

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 207 shows all attributes of TimeTariffInterval.

Table 207 – 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.
startDateTime	AbsoluteDateTime	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 208 shows all association ends of TimeTariffInterval with other classes.

Table 208 – 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..*] TariffProfiles	TariffProfile	All tariff profiles defined by this time tariff interval.

6.2.10.30 Transaction

The record of details of payment for service or token sale.

Table 209 shows all attributes of Transaction.

Table 209 – 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
description	String	inherited from: IdentifiedObject
localName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject
pathName	String	inherited from: IdentifiedObject

Table 210 shows all association ends of Transaction with other classes.

Table 210 – Association ends of PaymentMetering::Transaction with other classes

[mult from]	[mult to] name	type	description
[0..*]	[0..1] MeterAsset	MeterAsset	Meter asset for this vending transaction.
[0..*]	[0..1] AuxiliaryAccount	AuxiliaryAccount	Auxiliary account for this payment transaction.
[0..*]	[0..1] VendorShift	VendorShift	Vendor 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] CashierShift	CashierShift	Cashier shift during which this transaction was recorded.
[0..*]	[0..1] CustomerAccount	CustomerAccount	Customer account for this payment transaction.
[0..*]	[0..1] PricingStructure	PricingStructure	Pricing structure applicable for this 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.

6.2.10.31 Transactor

The entity that ultimately executes the transaction and who 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 211 shows all attributes of Transactor.

Table 211 – Attributes of PaymentMetering::Transactor

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

Table 212 shows all association ends of Transactor with other classes.

Table 212 – 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.

6.2.10.32 Vendor

The entity that owns PointOfSale and contracts with cashier to receipt payments and vend tokens using the payment system. Vendor has a private contract with and is managed by merchant who is a type of organisation. Vendor is accountable to merchant for revenue collected, who is in turn accountable to supplier.

Table 213 shows all attributes of Vendor.

Table 213 – Attributes of PaymentMetering::Vendor

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

Table 214 shows all association ends of Vendor with other classes.

Table 214 – 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.

6.2.10.33 VendorShift

The operating shift for a vendor during which he 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 215 shows all attributes of VendorShift.

Table 215 – 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	= true if 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
description	String	inherited from: IdentifiedObject
localName	String	inherited from: IdentifiedObject
mRID	String	inherited from: IdentifiedObject
name	String	inherited from: IdentifiedObject
pathName	String	inherited from: IdentifiedObject

Table 216 shows all association ends of VendorShift with other classes.

Table 216 – Association ends of PaymentMetering::VendorShift with other classes

[mult from]	[mult to] name	type	description
[0..1]	[0..*] Receipts	Receipt	
[0..1]	[0..*] Transactions	Transaction	
[0..*]	[0..1] Vendor	Vendor	Vendor that opens and owns this vendor shift.
[0..*]	[0..1] MerchantAccount	MerchantAccount	Merchant account this vendor shift periodically debits (based on aggregated transactions).

Bibliography

IEC 60050 (all parts), *International Electrotechnical vocabulary*

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 61968-9, *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 61970-552-4 *Energy management system application program interfaces – Part 552-4 CIM XML Model Exchange Format*[4)]

ISO 7812-1, *Identification cards – Identification of issuers – Part 1: Numbering system*

ISO 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*

4) Under consideration.

SOMMAIRE

COMMISSION ÉLECTROTECHNIQUE INTERNATIONALE

INTÉGRATION D'APPLICATIONS POUR LES SERVICES ÉLECTRIQUES – INTERFACES SYSTÈME POUR LA GESTION DE LA 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.

La présente version bilingue (2013-01), correspond à la version anglaise monolingue publiée en 2010-07.

Le texte anglais de cette norme est issu des documents 57/1058/FDIS et 57/1073/RVD.

Le rapport de vote 57/1073/RVD donne toute information sur le vote ayant abouti à l'approbation de cette norme.

La version française de cette norme n'a pas été soumise au vote.

Cette publication a été rédigée selon les Directives ISO/CEI, Partie 2.

Dans les sections informatives de ce document, les mots en police Tahoma gras s'appliquent aux termes qui sont utilisés comme jetons dans les articles normatifs (pour faciliter la lecture et la recherche dans le texte).

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 la distribution*, peut être consultée sur le site web de la CEI.

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 l'un avec l'autre en utilisant des logiciels intermédiaires (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 sont appropriées pour les applications faiblement couplées avec une plus grande hétérogénéité dans le langage, les systèmes d'exploitation, les protocoles et les 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 de logiciels intermédiaires, qui échangent des messages parmi des applications, complétera, ne remplacera pas, les centrales 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) se compose de divers composants d'application distribués qui ont pour mission de faire en sorte que l'entreprise de distribution contrôle les réseaux de distribution électriques. Ces fonctions 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 électrique, 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), qui est décrit dans CEI 61968-1.

INTÉGRATION D'APPLICATIONS POUR LES SERVICES ÉLECTRIQUES – INTERFACES SYSTÈME POUR LA GESTION DE LA DISTRIBUTION –

Partie 11: Extensions du modèle d'information commun (CIM) pour la distribution

1 Domaine d'application

La présente Norme internationale spécifie les extensions à la distribution du Modèle d'Information Commun (CIM) défini dans la CEI 61970-301. Il définit un jeu standard d'extensions du modèle d'information commun (CIM), qui soutient des définitions de messages dans les Parties 3 à 9 de la CEI 61968, la CEI 61968-13 et la CEI 61968-14[1)].Le domaine d'application de ce document est le modèle d'information qui étend la base CIM aux besoins des réseaux de distribution, ainsi qu'à l'intégration dans des systèmes d'information à l'échelles d'entreprise utilisé typiquement dans des compagnies d'électricité. Le modèle d'information est défini en UML, qui est un langage indépendant de la plateforme et programmable informatiquement, qui est utilisé pour créer des messages de définitions de la charge utiles dans différents formats requis. De cette façon, cette norme ne sera pas impactée par la spécification, le développement, et/ou le déploiement des architectures de futures générations, ou bien au travers de l'utilisation de normes ou de moyens propriétaires.

Pour les besoins de ce document, le modèle de distribution CIM (DCIM) se refère au modèle CIM du CE 57 de la CEI comme défini par la CEI 61970-301 et la CEI 61968-11 (le présent document).

Le Modèle d'Information Commun (CIM) est un modèle abstrait des objets principaux d'une entreprise 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 vendeurs. Le CIM facilite l'intégration en définissant un langage commun (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 CIM central pour les applications de Système de gestion de l'énergie (EMS), 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 CIM central par des paquetages qui sont axés sur les Systèmes de gestion de distribution (DMS) comprenant les Assets (les biens), Work (les travaux), Customers (les clients), Load Control (le contrôle de la charge), Metering (le mesurage) et autres. La CEI 61970-302 [2)] étend le CIM par des paquetages qui sont axés sur Financial (le financier), Energy Scheduling (la planification de l'énergie), Reservation (la Réservation) et d'autres applications relatives au marché. D'autres extensions du CIM peuvent être publiées sous forme de Normes internationales, qui sont chacune maintenues dans 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.

---

1) La CEI 61968-5, la CEI 61968-6, la CEI 61968-7, la CEI 61968-8 et la CEI 61968-14 sont à l'étude.

2) A l'étude.

2 Références normatives

Les documents de référence suivants sont indispensables pour l'application du présent document. 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 61968-1, *Application integration at electric utilities – System interfaces for distribution management – Part 1: Interface architecture and general requirements* (disponible en anglais seulement)

CEI 61968-2, *Application integration at electric utilities – System interfaces for distribution management – Part 2: Glossary* (disponible en anglais seulement)

CEI 61970-301, *Interface de programmation d'application pour systèmes de gestion d'énergie (EMS-API) – Partie 301: Base de modèle d'information commun (CIM)*

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 seulement)

3 Termes et définitions

Pour les besoins du présent document, les termes et définitions donnés de la CEI 61968-2 et suivants s'appliquent.

NOTE Se référer au Vocabulaire Électrotechnique International, CEI 60050, pour les définitions du glossaire général.

3.1
système de gestion d'énergie
EMS (*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 (*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

3.3
langage de modélisation unifié
UML (*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

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 (*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

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

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 (langage de balisage extensible, *extensible markup language*)

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 (*resource description framework*)
la norme web (W3C) utilisée pour représenter les modèles d'informations

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 (*web ontology language*)
autre norme Web (W3C) qui inclut RDF et l'étend

Le langage OWL est plus puissant que RDF pour la prise en charge des types de données, des énumérations, de 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) 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 modèle d'information commun 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 du CE 57

Le CIM complet est réparti en groupes de paquetages pour des raisons de commodité. Ces groupes 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 CEI 61968-11 (le présent document).

La Figure 1 montre tous les paquetages normatifs CIM du CE 57 de la CEI 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.

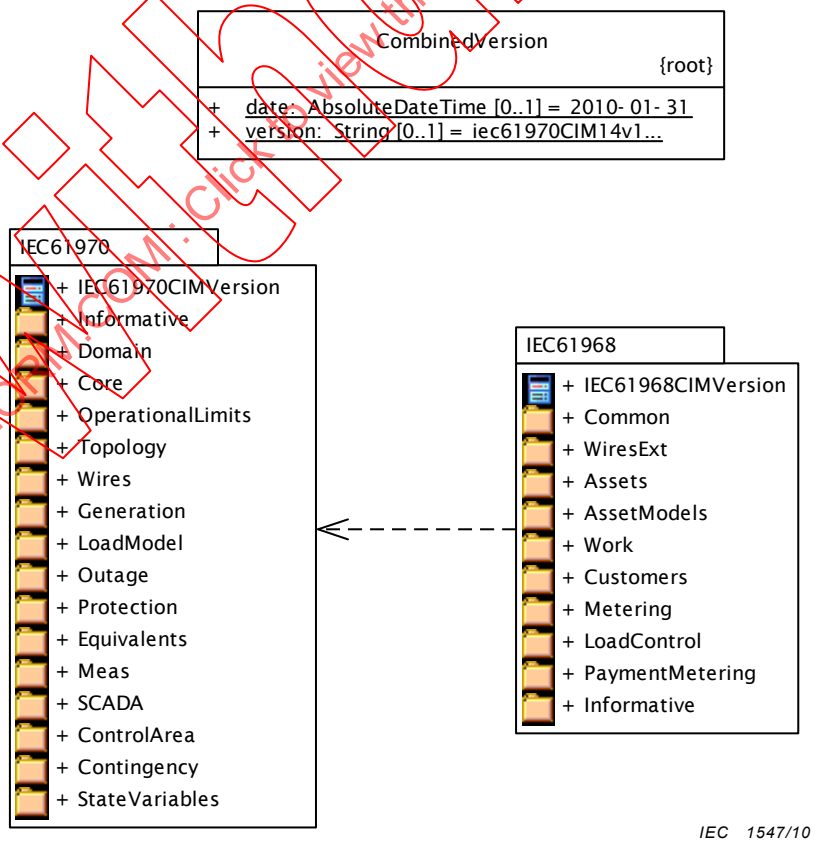


Figure 1 – Paquetages CIM du CE 57

NOTE Le contenu du CIM de base auquel il est fait référence issu de la présente spécification a été généré automatiquement à partir de la version de modèle électronique UML du CIM de base IEC61970CIM14v13.

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 **Wires** (les fils), **Topology** (la topologie), **Measurements** (les mesures), **Equivalents** (les équivalents) et **Core** (le cœur ou noyau), ainsi que plusieurs autres. Les parties 3 à 9 de la CEI 61968, la CEI 61968-13 et la CEI 61968-14 exigeaient des extensions apportées au modèle CIM tel que 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 contrôle, mais aussi aux systèmes d'entreprises et partenaires. 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.

La Figure 2 montre les paquetages définis pour la CEI 61968-11 Extension du CIM pour la distribution. 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 61968, ainsi que différentes applications qui utilisent le CIM pour l'échange d'informations, définiront typiquement des messages en utilisant des classes issues de plusieurs paquetages, notamment certaines qui sont définies hors du présent document.

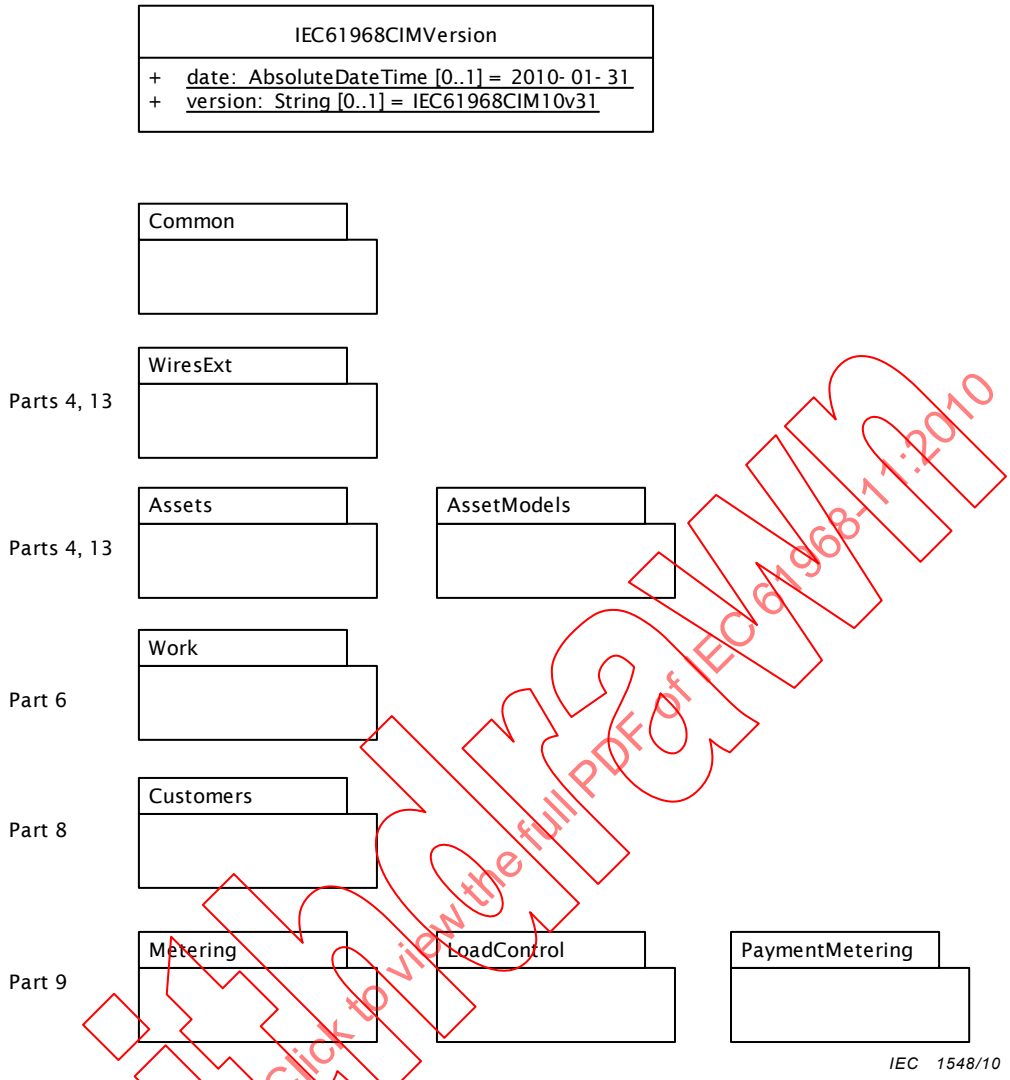


Figure 2 – Paquetages de haut niveau d'extensions CIM pour la distribution (DCIM)

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 IEC61968CIM10v31.

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 outils comme un CIMTool pour générer des messages spécifiques au contexte au format XSD, RDF ou OWL pour les parties 3 à 9 de la série de normes CEI 61968, CEI 61968-13 et CEI 61968-14. D'autres ensembles d'outils peuvent aussi être utilisés.

Le présent paragraphe (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 au Paragraphe 4.3 de la CEI 61970-301: Base de Modèle d'Information Commun (CIM)

4.3.2 Domaine d'application du modèle UML

La présente spécification 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 parties 3 à 9 de la série de normes CEI 61968, CEI 61968-13 et CEI 61968-14. Les extensions personnalisées constituent le sujet de projets et de produits non normalisés. Elles peuvent être utilisées pour exercer un effet de levier 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 évolue depuis de nombreuses années, mais n'a jamais été publié sous forme d'une norme CEI. 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 61968-9 et la CEI 61968-13. 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 parties 3 à 9 de la série de normes CEI 61968 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 11 et pour la 2e 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 et CEI 61968-13.

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 nommage local. L'espace de nommage 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 «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 nommage cible doit être différent d'un espace de nommage XSD du CIM afin d'identifier l'origine de l'extension.

L'espace de nommage de l'extension peut suivre une convention de nommage: http://<organisation>/<contrôle de version>/<profil>.

4.3.4 Définition (profil) 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 du message à partir du modèle CIM (CEI 61968-11 et CEI 61970-301), du format d'en-tête et 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 messages normalisés (profils) consiste à utiliser des applications telles que l'outil "open source" (de source ouverte) CIMTool. D'autres méthodes peuvent être utilisées, y compris le codage manuel.

A l'heure actuelle, deux grammaires du XML sont utilisées pour la définition des profils dans la série CEI 61968.

4.3.4.2 Utilisation du schéma XML

Les messages définis dans les parties 3 à 9 de la série de normes CEI 61968 utilisent des combinaisons de noms et de verbes. Les noms renvoient en général aux classes définies au sein du modèle DCIM.

Le verbe indique en général les attributs qui sont requis. Dans le cas des verbes **CREATE/CREATED** et **SHOW**, tous les attributs définis dans un profil sont requis, alors que dans le cas des verbes **GET**, **CANCEL/CANCELLED**, **DELETE/DELETED** et **CLOSE/CLOSED**, seuls les identificateurs d'objet sont typiquement requis. Le verbe **CHANGE/CHANGED** exige un identificateur d'objet et les valeurs des attributs à changer.

Les verbes définis pour être utilisés dans les interfaces conformes à la CEI 61968 sont spécifiés dans la CEI 61968-1.

4.3.4.3 Utilisation 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 61970-552-4[3] et sont actuellement utilisées par les profils définis dans la CEI 61968-13.

4.4 Concepts du modèle CIM et exemples

4.4.1 Généralités

Le présent paragraphe (4.4) décrit un certain nombre d'exemples de modélisation dans le domaine de la distribution avec DCIM. Ceux-ci sont complémentaires des exemples du CIM de base présentés dans la CEI 61970-301.

4.4.2 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, ainsi que (b) l'échange d'informations relatif aux opérations du réseau et à la planification dans le contexte de l'ensemble de l'entreprise de distribution d'électricité.

La Figure 3 montre un certain nombre de classes-clés dans le DCIM.

---

3 A l'étude.

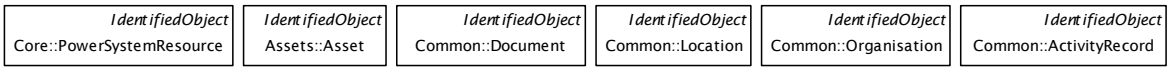


Figure 3 – Classes-clés du DCIM

Un **Asset** (un Bien) est une ressource tangible de l'entreprise de services publics, y compris les équipements des systèmes de puissance, les outils, les armoires, les bâtiments, etc. Pour les équipements du réseau électrique, le rôle d'un bien est défini par le truchement de la hiérarchie PowerSystemResource, définie principalement dans le modèle Wires (se référer à la CEI 61970-301 et au paquetage modèle **IEC61970::Wires**, et le paquetage de modèle **IEC61968::WiresExt** dans l'Article 6). La description du bien met l'accent sur les caractéristiques physiques de l'équipement remplissant le rôle en question.

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 de puissance.

Un **Location** (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.

Les **Organisations** 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.

NOTE 1 Les classes du CIM sont souvent utilisées dans un grand nombre de contextes disparates. Par exemple, la même instance de Organisation peut concerner le rôle de client dans un contexte traitant de demandes de services et le rôle d'un fabricant dans un contexte différent traitant de la maintenance d'un bien particulier.

NOTE 2 Plusieurs de ces classes-clés ont la relation à d'autres classes (non montrée dans la Figure) et il existe en particulier des relations entre leurs sous-classes. Par exemple, un **Asset** joue un rôle particulier dans le réseau électrique tel que défini par le **PowerSystemResource** auquel il est associé. Le **PowerSystemResource** peut avoir un emplacement schématique sur un schéma unifilaire et un emplacement géographique, alors que l'**Asset** remplissant ce rôle a un emplacement géographique qu'une équipe peut desservir. Si un **Asset** est retiré du service, remis en état et ensuite installé en un autre **Location**, l'historique du réseau logique et les biens physiques peuvent tous deux être suivis à la trace en ayant des instances de **ActivityRecord** associées à **PowerSystemResource** ainsi qu'à **Asset**. Le **PowerSystemResource** peut servir un **Organisation** particulier, tel qu'un fournisseur. Ce fournisseur est simultanément un client.

4.4.3 Charges monophasées et déséquilibrées

La Figure 4 montre les classes disponibles pour modéliser les charges de distribution, qui sont souvent déséquilibrées parmi les trois phases en un emplacement. Dans certains cas, il se produit des charges monophasées et diphasées.

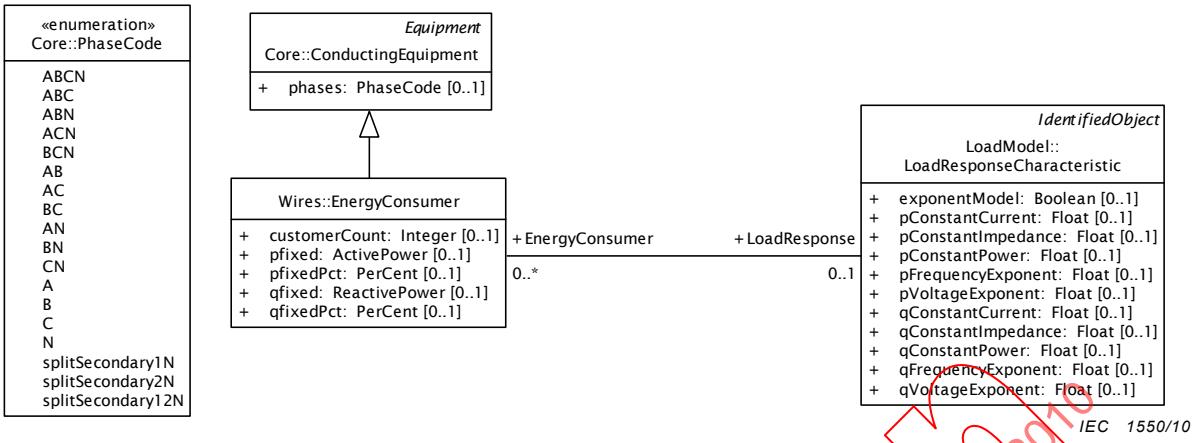


Figure 4 – Modèle de charge du DCIM

Il convient d'utiliser la classe **EnergyConsumer** pour instancier le modèle de charges, qui peut facultativement être associé à un compteur (par le truchement de **ServiceDeliveryPoint**, non montré dans la figure), mais peut ne pas l'être. **EnergyConsumer** hérite de l'attribut **phases** de **ConductingEquipment**, 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.

Si une charge triphasée est déséquilibrée, elle peut être modélisée par trois charges monophasées reliées (par des **Terminal**, des bornes) au même **ConnectivityNode**. Avec les charges monophasées ou diphasées, il convient que le modèle inclue des conducteurs ou des transformateurs qui établissent la connectivité de retour à la source.

Le modèle de charge pourrait être quelque combinaison de courant constant, de puissance constante ou d'impédance constante. Dans ce cas, il convient d'associer une instance de **LoadResponseCharacteristic** à l'instance de **EnergyConsumer**.

4.4.4 Segments de ligne de distribution

La Figure 5 montre les classes disponibles pour modéliser les segments de ligne de distribution (des conducteurs). Il y a quatre façons de décrire les paramètres d'impédance d'un **DistributionLineSegment**, en fonction du type d'échange de modèle et des données disponibles dans le système d'exportation.

NOTE La présente édition de la CEI 61968-11 (documentant le modèle DCIM10) et l'édition correspondante de la CEI 61970-301 (document le modèle CIM14 de base) reflètent respectivement des modèles distincts de segment de ligne pour la transmission et la distribution (T&D). Il convient que les prochaines éditions présentent un modèle T&D consolidé de segment de ligne.

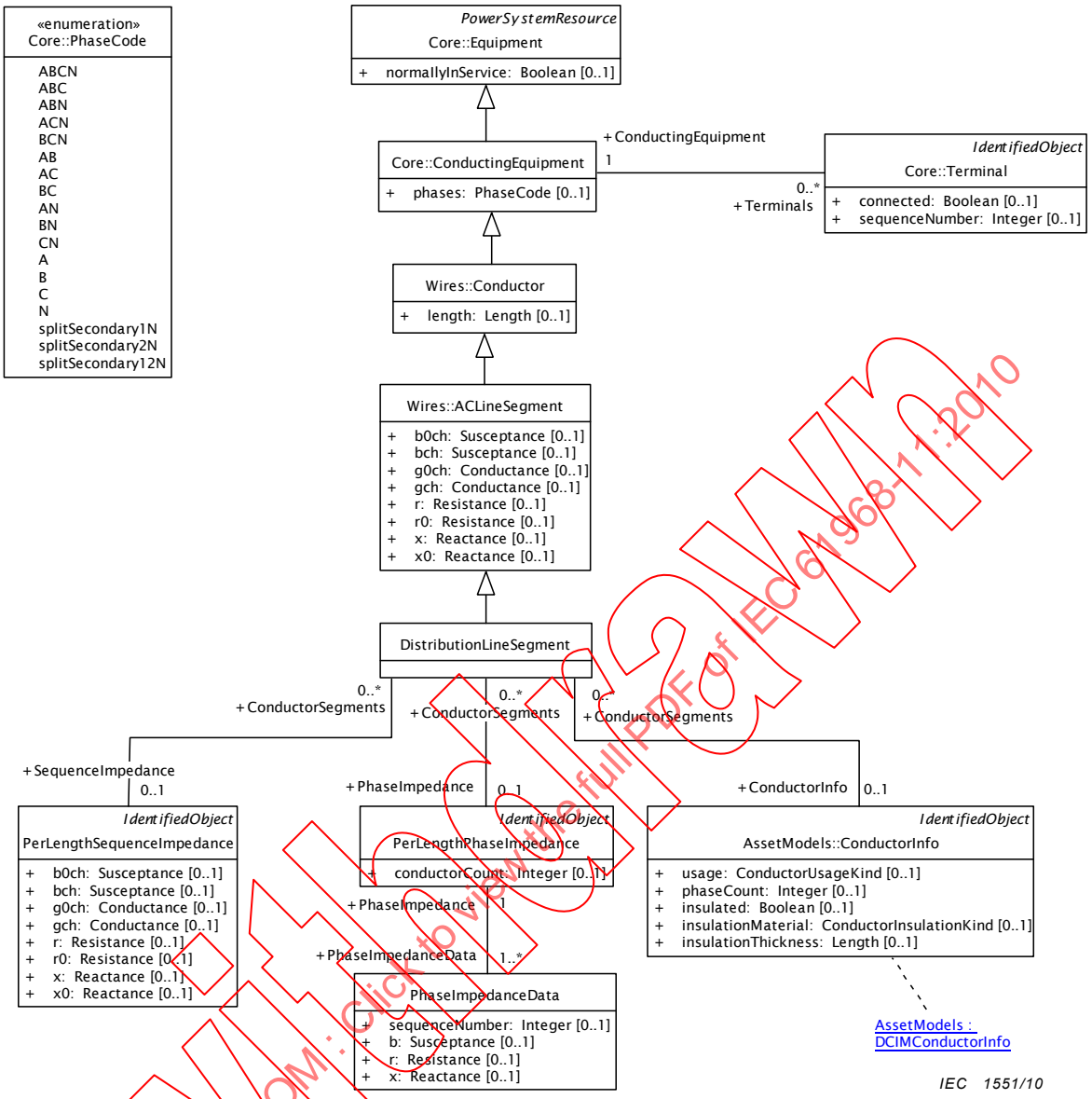


Figure 5 – Modèle de connectivité de lignes DCIM

- Pourvoir une association à **ConductorInfo**, détaillée à la Figure 6, pour le calcul de l'impédance et de l'admittance linéiques à partir des équations de Carson en utilisant le câble physique et les données géométriques. L'attribut **Conductor.length** (la longueur de conducteur) est requis car les paramètres électriques de l'instance sont définis comme étant (paramètres linéiques) * (**Conductor.length**). Voir aussi 4.4.4.3 ci-dessous.

- Pourvoir une association à **PerLengthPhaseImpedance**, qui fait référence aux matrices symétriques d'impédance et d'admittance NxN précalculées par unité de longueur. Le **conductorCount** (*N*) 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 (instances de **PhaseImpedanceData**). L'énumération **PhaseCode** n'étant pas ordonnée, les rangées et les colonnes de la matrice doivent être dans l'ordre des phases. Pour **phases=ABC**, la rangée 1 est toujours la phase **A**, la rangée 2 est la phase **B** et la rangée 3 est la phase **C**. Cela pourrait décrire un «code de ligne» avec la phase **A** à gauche, **B** au milieu et **C** à droite. Si le phasage est **BCA**, de gauche à droite, cela exigerait une instance différente pour **PerLengthPhaseImpedance** car les

matrices différeront. **Conductor.length** est requis pour le **DistributionLineSegment**. Voir aussi 4.4.4.2 ci-dessous.

- Pourvoir une association à **PerLengthSequenceImpedance**. Cette classe met en œuvre une bibliothèque de «codes de ligne» qui ont des impédances en séquence et un chargement de ligne linéique. **Conductor.length** est requis. Voir 4.4.4.2 pour l'usage des paramètres séquentiels avec des segments de ligne monophasée et diphasée.
- Les attributs hérités **ACLineSegment** peuvent être utilisés si aucune des trois associations précédentes n'est fournie. **Conductor.length** est facultatif et n'est pas utilisé dans le calcul d'impédance.

La Figure 6 montre la hiérarchie de la classe **ConductorInfo**, utilisée pour définir les impédances de segment de ligne à partir de données physiques (parfois appelées «codes de ligne»). **ConductorInfo** est une classe de base qu'il convient de ne pas instancier directement.

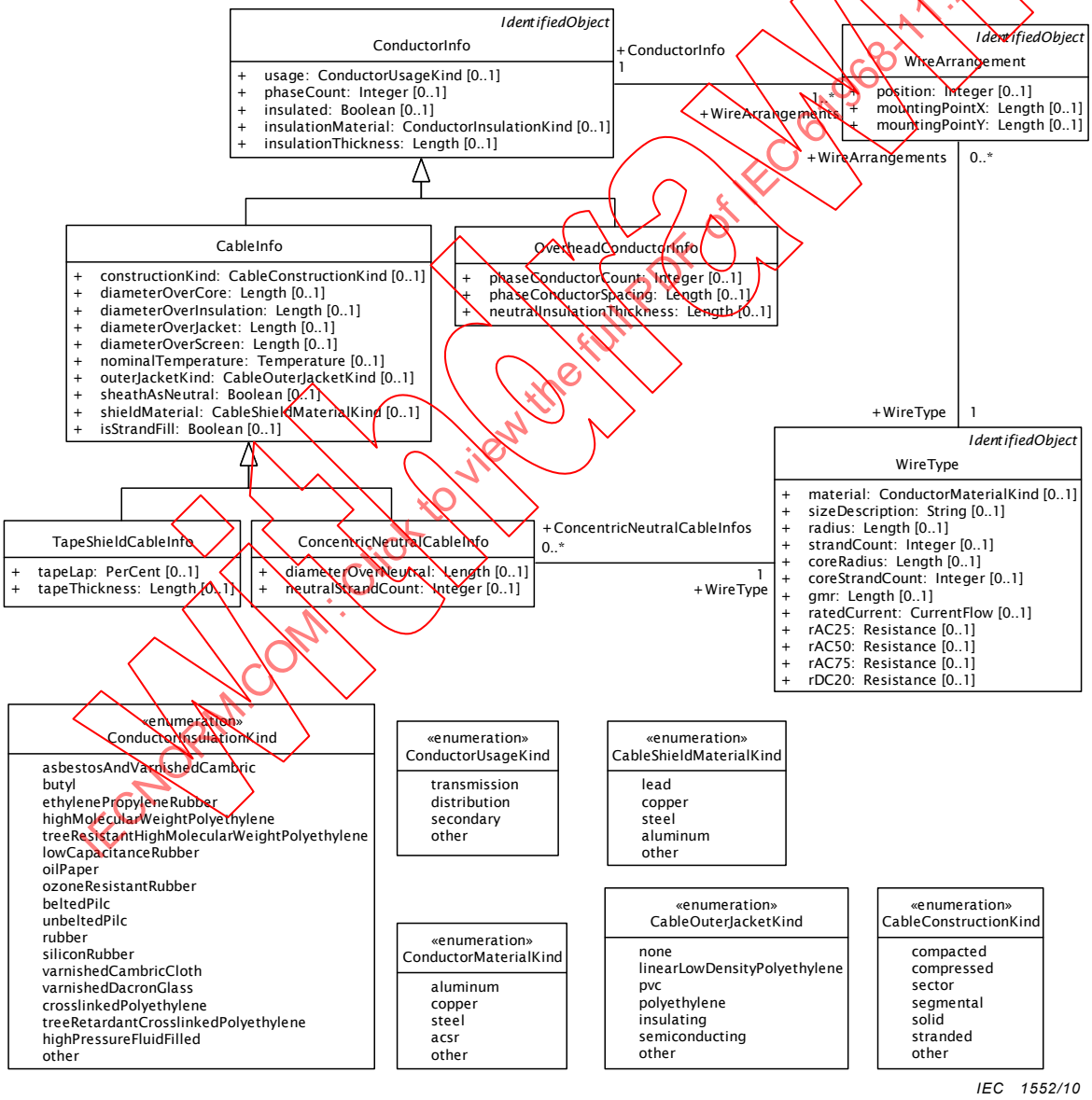


Figure 6 – Modèle (feuille de données sur les lignes et les câbles) de conducteurs DCIM

OverheadConductorInfo identifie les données physiques de ligne. Tous les éventuels conducteurs ayant un numéro supérieur à **ConductorInfo.phaseCount** 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. Les attributs **phaseConductorCount** et

phaseConductorSpacing de **OverheadConductorInfo** 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. **OverheadConductorInfo** prend aussi en compte les lignes secondaires triplex, en spécifiant les attributs **insulated**, **insulationMaterial**, **insulationThickness** et **neutralInsulationThickness**.

WireType définit une bibliothèque de câbles. Les attributs **material**, **sizeDescription**, **strandCount** et **coreStrandCount** (pour l'ACSR, à savoir "aluminum conductor steel reinforced wire", fil conducteur en aluminium renforcé d'acier) 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 de flux de puissance, mais pas pour la solution du flux de puissance elle-même.

WireArrangement définit les coordonnées horizontale (**mountingPointX**) et verticale (**mountingPointY**) de chaque fil sur la section de pôle. Tous les trois attributs sont requis. La hauteur du fil par rapport au sol est **mountingPointY**, y compris tous les éventuels effets de flèche. La position horizontale du fil, **mountingPointX**, 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. La classe **WireArrangement** intercale une relation plusieurs à plusieurs entre **ConductorInfo** et **WireType**. Ce modèle permet d'avoir un **WireType** différent pour le neutre et pour chaque phase.

Les câbles souterrains utiliseront **WireArrangement** et **WireType** de la même que le font les câbles aériens. Les deux classes concrètes qui peuvent être instanciées sont **TapeShieldCableInfo** et **ConcentricNeutralCableInfo**. Pour les câbles à neutre concentrique, les fils de neutre de cuivre sont définis par une association à **WireType**, leur nombre et leur positionnement étant déterminés par les attributs **diameterOverNeutral** et **neutralStrandCount**.

4.4.4.1 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 **DistributionLineSegment**. Les attributs **bch**, **b0ch**, **gch** et **g0ch** ne sont 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 Ceci équivaut aux attributs de **Wires::ACLineSegment**, qui sont précalculés pour la longueur totale du segment et doivent donc être 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 **DistributionLineSegment** a seulement une ou deux phases, un modèle équilibré peut encore être transféré à travers les attributs **r**, **x**, **r0** et **x0**.Ceci 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 d'attribuer à l'attribut hérité **phases** de **DistributionLineSegment** un libellé d'énumération **PhaseCode** approprié pour indiquer les phases réellement présentes, tel que **A**, **B**, **C**, **AB**, **BC**, **AC** ou **ABC**. 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**.

4.4.4.2 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 4.4.4.3. Un modèle de matrice est utile lorsque:

- l'application cible n'a aucun moyen de calculer le 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 rangée 1, colonne 1 de la matrice
- **sequenceNumber** = 2 pour la rangée 2, colonne 1 de la matrice
- **sequenceNumber** = 3 pour la rangée 2, colonne 2 de la matrice

Cette instance de **PerLengthPhaseImpedance** pourrait être référencée à partir d'un **DistributionLineSegment** ayant les phases **AB**, **AC** ou **BC**. La rangée 1 correspond toujours à la première phase, qui est **A**, **B**, ou **C**. Si les fils de phase sont transposés physiquement, et il convient de placer **C** (par exemple) dans la rangée 1, une nouvelle instance **PerLengthPhaseImpedance** est requise.

4.4.4.3 Utilisation des paramètres physiques

Pour les lignes aériennes, un modèle physique peut être transféré par référence à une instance de **OverheadConductorInfo**, qui est associée à des classes supplémentaires montrées à la Figure 6. Cela viendra à l'appui du calcul d'une matrice des impédances non équilibrées par le truchement d'une 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 **OverheadConductorInfo**, quatre instances de **WireArrangement** qui décrivent les quatre positions des conducteurs, et deux instances de **WireType** décrivant les types des fils de phase et du neutre. L'attribut **length** de **Conductor** doit être utilisé, et plusieurs **DistributionLineSegment** se réfèreront typiquement à la même instance de **OverheadConductorInfo**.

En plus de définir les points de montage, les instances **WireArrangement** relient les **WireType** et les **OverheadConductorInfo**. Ces **WireArrangement** doivent être séquencés afin d'identifier les attributions de fils de phase. Pour les calculs, il convient de fournir l'attribut résistance de **WireType** pour la fréquence du réseau et à la température de fonctionnement souhaité du fil.

Un câble à neutre concentrique monophasé requiert une instance de **ConcentricNeutralCableInfo**, une instance de **WireArrangement** pour spécifier la profondeur d'enfouissement, et un **WireType** pour le conducteur central. Un second **WireType** spécifie le brin neutre; il est associé à **ConcentricNeutralCableInfo**, directement plutôt que par l'intermédiaire d'un **WireArrangement**. Un câble blindé triphasé, avec un conducteur neutre nu, exigerait une instance de **TapeShieldCableInfo** et quatre **WireArrangement** pour les trois phases et le neutre. Il y aurait également deux **WireType**, un pour le conducteur de phase central et l'autre pour le neutre.

4.4.5 Transformateurs de distribution

4.4.5.1 Modèle électrique

La Figure 7 montre les classes qui modélisent les instances du transformateur de distribution. Elles s'appuient exclusivement sur une bibliothèque de types de transformateur, détaillée à la Figure 8, pour obtenir la plupart des paramètres de transformateur.

NOTE La présente édition de la CEI 61968-11 (documentant le modèle DCIM10) et l'édition correspondante de la CEI 61970-301 (document le modèle CIM14 de base) reflètent respectivement des modèles distincts de transformateur pour la transmission et la distribution (T&D). Il convient que les prochaines éditions présentent un modèle T&D consolidé de transformateur.

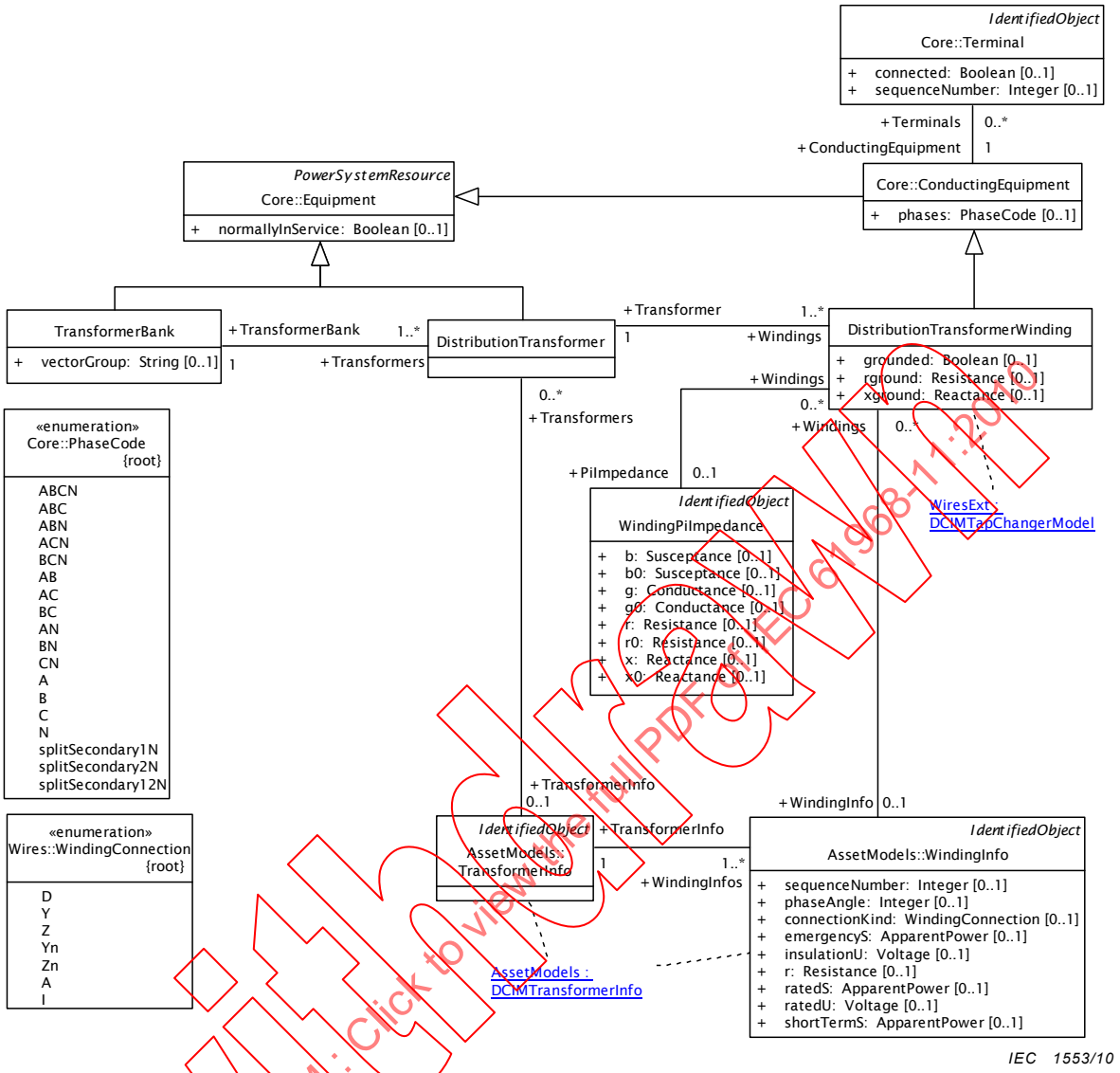


Figure 7 – Modèle de connectivité des transformateurs DCIM

TransformerBank organise les transformateurs constitutifs triphasés et/ou monophasés. Certaines batteries contiennent juste un seul transformateur triphasé, alors que d'autres contiennent deux ou plusieurs transformateurs monophasés. 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. Cette classe est l'équivalent de **PowerTransformer** défini dans la CEI 61970-301 et elle hérite de **Equipment**. Des emplacements sont souvent associés à la batterie. L'attribut **vectorGroup** pour le relaiement protecteur est dérivé des connexions d'enroulements internes et des angles de phase. Il utilise la nomenclature normalisée pour décrire le nombre des enroulements qui peuvent être compris dans la batterie.

DistributionTransformer doit résider dans une batterie de transformateurs et il organise un nombre flexible d'enroulements de transformateur. Il fait référence à la classe **TransformerInfo** dans la Figure 8 pour obtenir la plupart des données.

DistributionTransformerWinding est un **ConductingEquipment**, avec les informations relatives au phasage et des **Terminal** associés. Il fait référence à la classe **WindingInfo** dans la Figure 8 pour obtenir des données relatives aux caractéristiques assignées des enroulements. En

plus de **ConductingEquipment.phases**, 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.

La classe **WindingPiImpedance** permet la compatibilité avec les modèles de transformateur de la CEI 61970-301.

4.4.5.2 Modèle physique

La Figure 8 montre les classes qui permettent l'échange de données de physiques des transformateurs, parfois appelées «codes de transformateur» dans les applications.

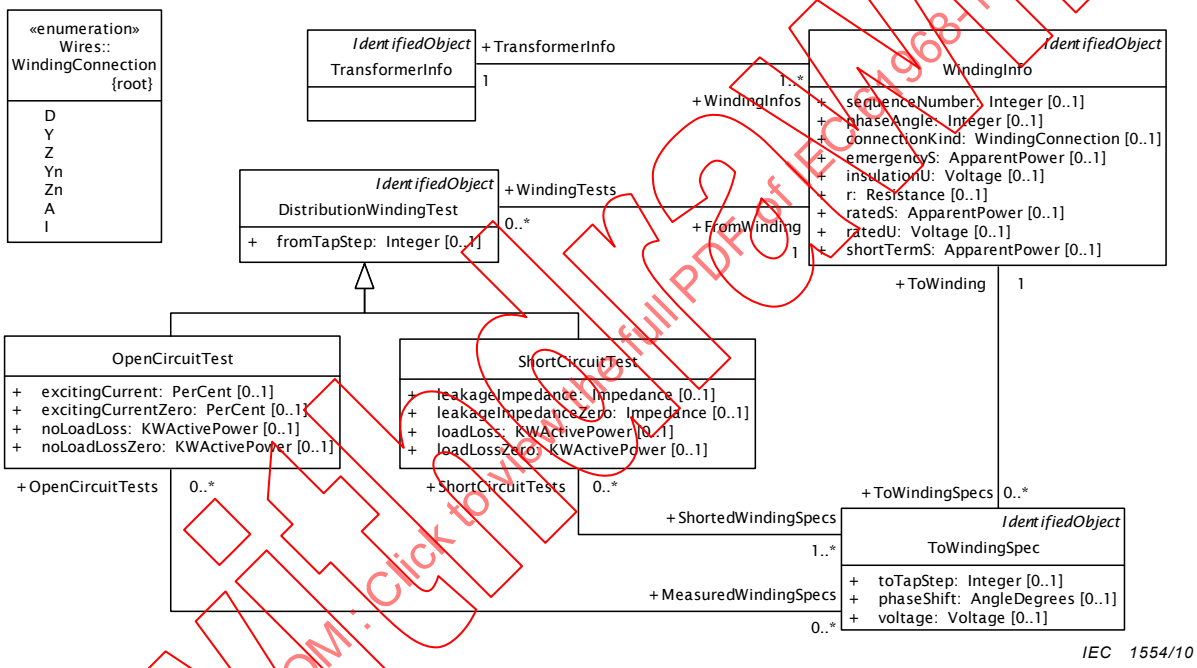


Figure 8 – Modèle de feuilles de données des transformateurs DCIM

L'énumération **WindingConnection** comprend la nomenclature normalisée **D**, **Y**, **Z**, **Yn** et **Zn** pour décrire les connexions delta, étoile (en Y), zig-zag 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.

TransformerInfo est référencé par **DistributionTransformer**. Il définit la plupart des données de transformateur. Dans le modèle actuel, cette classe n'a pas d'attributs propres et sert seulement à organiser les attributs d'enroulement en une bibliothèque. **TransformerInfo** rassemble les associations à **WindingInfo**, permettant ainsi de ne pas utiliser l'énumération **WindingType**. Cela se généralise à n'importe quel nombre d'enroulements.

WindingInfo inclut toutes les informations relatives aux caractéristiques assignées des enroulements, y compris **ratedU**, **ratedS**, **connectionKind**, **phaseAngle**, **shortTermS**, **emergencyS**, et **insulationU**. **r** est la résistance CC de l'enroulement. **sequenceNumber** est le numéro d'ordre de l'enroulement dans le groupe **vectorGroup**; l'enroulement haute tension est toujours 1. **sequenceNumber** augmente à mesure que **ratedU** diminue, mais certains transformateurs comportent deux enroulements ayant le même **ratedU**, auquel cas **sequenceNumber** établit la distinction entre eux.

DistributionWindingTest avec ses attributs et sous-classes couvre les essais de court-circuit et de circuit ouvert, pour les transformateurs monophasés ou triphasés, et dans le cas des transformateurs triphasés, il couvre l'excitation soit directe, soit homopolaire. Pour les essais de circuit ouvert et les essais de court-circuit, un seul enroulement (“from”) est excité. L'enroulement “from” winding est associée à la classe parente **DistributionWindingTest** . Pour les essais de court-circuit, un ou plusieurs enroulements (“to”) seront mis en court-circuit. Pour les essais de circuit ouvert, il peut y avoir des données relatives à la tension et aux angles sur les autres enroulements “to”.

Dans le cas de **OpenCircuitTest**, ses attributs **excitingCurrent**, **excitingCurrentZero**, **noLoadLoss** et **noLoadLossZero** sont tous mesurés sur l'enroulement (“from”) excité. Les données d'essai directes et homopolaires peuvent être rapportées dans la même instance de **OpenCircuitTest**. Ces essais sont en général réalisés avec la tension assignée appliquée à l'enroulement excité. L'essai peut inclure des mesures de la tension et de l'angle induits sur les autres enroulements. ces valeurs seraient rapportées dans des instances associées de **ToWindingSpec**.

ShortCircuitTest est réalisé en faisant circuler le courant assigné à travers l'enroulement “from”, avec un ou plusieurs enroulement “to” mis en court-circuit. Les données d'essai directes et homopolaires peuvent être rapportées dans la même instance de **ShortCircuitTest.** Les résistances CA dérivées de **loadLoss** et de **loadLossZero** vont vraisemblablement différer des résistances CC des enroulements, **WindingInfo.r**, qui sont obtenues à partir d'un essai distinct.

La classe **ToWindingSpec** définit les enroulements en court-circuit pour un **ShortCircuitTest** donné et seul l'attribut **toTapStep** s'applique. Pour un **OpenCircuitTest**, les attributs **voltage** et **phaseShift** contiennent la tension et l'angle mesurés sur l'enroulement “to” associé. Les instances de cette classe doivent être associées soit à un **OpenCircuitTest**, soit à un **ShortCircuitTest**, mais pas aux deux.

4.4.5.3 Modèle de changement de prise

La Figure 9 montre les classes utilisées pour modéliser un régulateur de tension déséquilibrée. Un **DistributionTapChanger**, qui hérite de **RatioTapChanger,** est associé à un **DistributionTransformerWinding**. Chaque régulateur utilise une commande locale autonome et, de ce fait, **RegulationSchedule** et **TapSchedule** (présents dans la CEI 61970-301 pour les transformateurs de puissance) ne sont pas utilisés. Certains attributs importants de ces classes ont été copiés dans **DistributionTapChanger**. Les régulateurs d'angle de phase et les courbes de variations ne sont pas également utilisés d'une manière générale dans les systèmes de distribution.

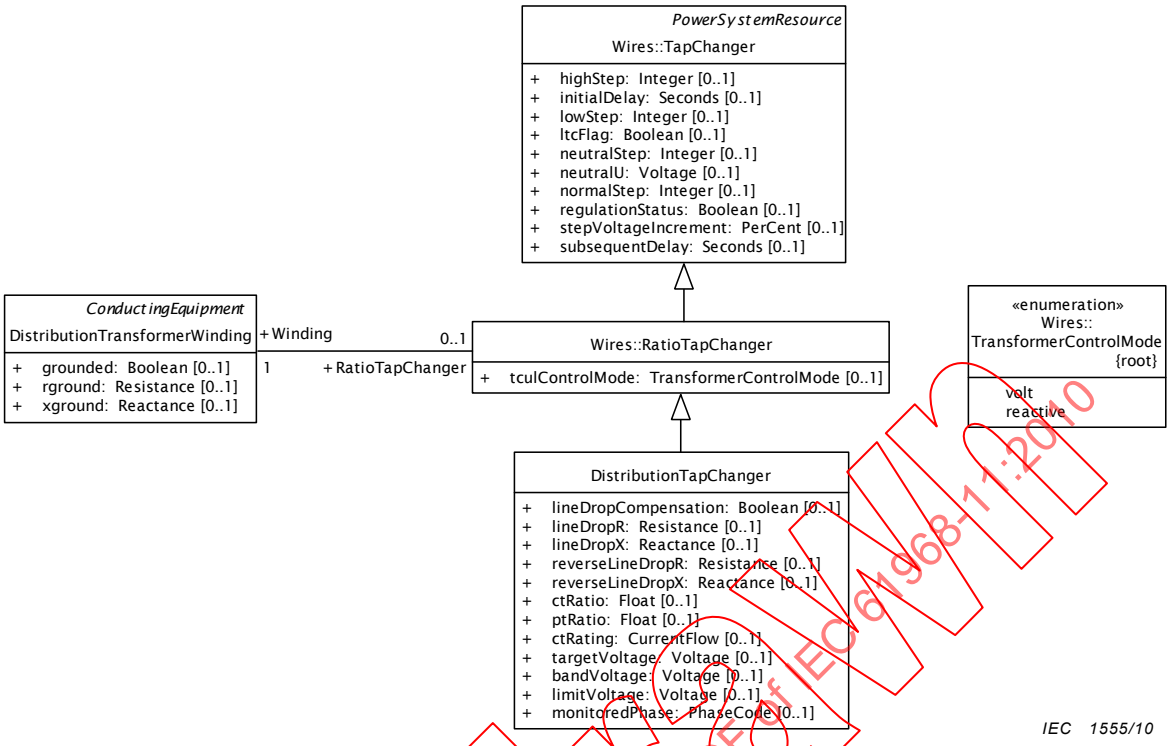


Figure 9 – Modèle de changeur de prise DCIM

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 **TransformerBank** contenant trois **DistributionTransformers** et un total de six **DistributionTransformerWinding**. Il y aura trois instances de **DistributionTapChanger**, associées chacune à un **DistributionTranformerWinding** différent. Il convient que l'attribut **monitoredPhase** de **DistributionTapChanger** soit adapté à la connexion des phases de son enroulement associé. 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 batterie, 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. Cela pourrait être modélisé avec une batterie d'un transformateur triphasé et d'un total de trois **DistributionTransformerWinding** triphasés. Il y aurait juste un seul **DistributionTapChanger** associé à un enroulement. L'attribut **monitoredPhase** de **DistributionTapChanger** 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 le régulateur.

4.4.5.4 Exemple de transformateur de distribution

Le transformateur dans la Figure 10 est une batterie étoile ouverte/triangle ouvert, qui est utilisée pour alimenter un service triphasé peu coûteux à des clients plus petits.

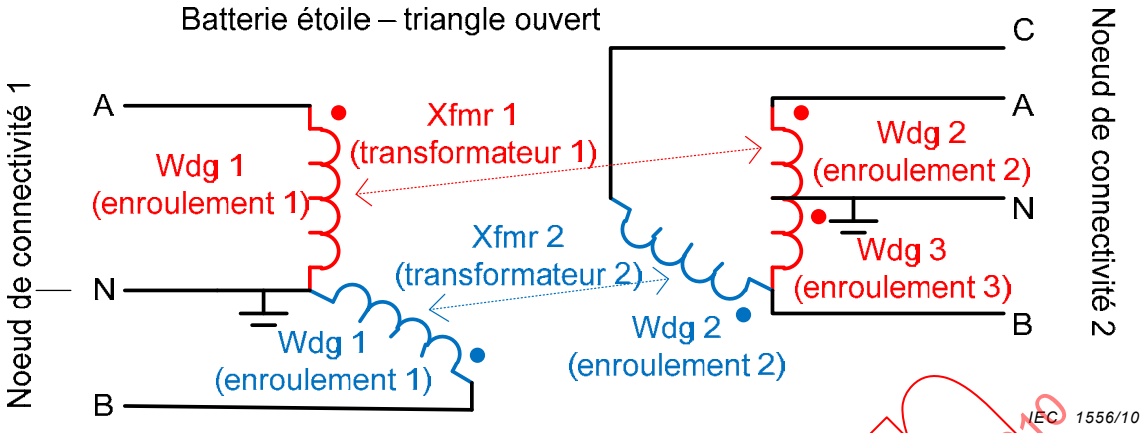


Figure 10 – Exemple de transformateur de distribution pouvant être modélisé par le DCIM

Le Tableau 1 montre certaines des valeurs d'attribut importantes pour cet exemple.

Tableau 1 – Connexions de batterie de transformateur en étoile ouverte / triangle ouvert

Transformateur de distribution	Enroulement de transformateur de distribution	ratedU	ratedS	Type de connexion	phaseAngle	phaseCode
Xfmr 1	Wdg 1	7200	100e3	I	0	AN
	Wdg 2	120	50e3	I	0	AN
	Wdg 3	120	50e3	I	6	BN
Xfmr 2	Wdg 1	7200	50e3	I	0	BN
	Wdg 2	240	50e3	I	0	BC

Un déphasage de “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** fournies par la batterie de transformateurs. Un autre équipement raccordé, tel qu'un segment de ligne, pourrait ajouter une phase **C**. **ConnectivityNode** 2 aura la présence des phases **ABCN** fournies par cette batterie de transformateurs. la “patte d'éclairage” (Xfmr 1) a habituellement une caractéristique assignée différente de celle de la “patte alimentation” (Xfmr 2). Cela signifie que les affectations de phases et de caractéristiques assignées à la batterie pourraient être ambiguës et il serait donc nécessaire de les spécifier sur **DistributionTransformerWinding**.

4.4.6 Connectivité avec des phases déséquilibrées

4.4.6.1 Généralités

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, 4.4.2 Connecitivity Model). **ConductingEquipment** a un attribut **phase** pour décrire les informations de phase relatives à l'équipement conducteur. Les **ConnectivityNode** sont les points où les bornes **Terminal** de **ConductingEquipment** sont reliées ensemble avec une impédance nulle. Ils *n'ont pas* d'informations relatives à la phase. La relation entre un **ConnectivityNode** et son conteneur est modélisée par le truchement de la classe **ConnectivityNodeContainer**.

NOTE Le modèle de connectivité et de phasage CIM présenté prend en charge l'échange de modèle de réseau définitif. Cependant, dans les opérations du réseau de distribution, il existe des cas où un

ConductingEquipment a un état différent pour chaque **phase**. Par exemple, un commutateur triphasé peut être actionné différemment sur chaque phase afin de débrancher électriquement la phase **A** et de brancher électriquement les phases **BC**; il en résulte que la ligne reliée au commutateur peut aussi bien avoir un état électrique différent sur chaque phase. Actuellement, le modèle de mesure CIM *ne peut pas* représenter ces cas car aucune information de phase n'est fournie sur **Measurement**. Cette capacité est bien nécessaire pour la commande et le fonctionnement de la distribution et peut être présente dans la prochaine édition de la présente CEI 61968-11 en appui à la CEI 61968-3.

Pour un exemple élaboré du réseau exemplaire triphasé équilibré, se référer à la CEI 61970-301, Paragraphe 4.4.2.1 «Exemple de connexité et d'emboîtement». Ces exemples fournis dans le prochain paragraphe illustrent l'utilisation du modèle de connectivité dans le cas des phases déséquilibrées.

4.4.6.2 Exemples de connectivité pour les phases déséquilibrées

D'une façon générale, un **ConnectivityNode** est créé comme étant le point où deux ou plusieurs dispositifs sont reliés ensemble. Les dispositifs, qui sont modélisés comme des sous-types de **ConductingEquipment**, transportent les informations relatives au phasage par le truchement de l'attribut **phase** de **ConductingEquipment**.

Considérer comme exemple **ACLineSegment** ou son sous-type **DistributionLineSegment**, qui sont utilisés pour modéliser la connectivité et les caractéristiques (impédances) électriques des lignes et câbles aériens. Il est nécessaire de modéliser électriquement la ligne ou le câble physique de sorte qu'un **ACLineSegment** ou **DistributionLineSegment** ait un et un seul attribut **phase**.

La Figure 11 illustre trois exemples avec différentes configurations de phases qui sont communs dans les réseaux de distribution.

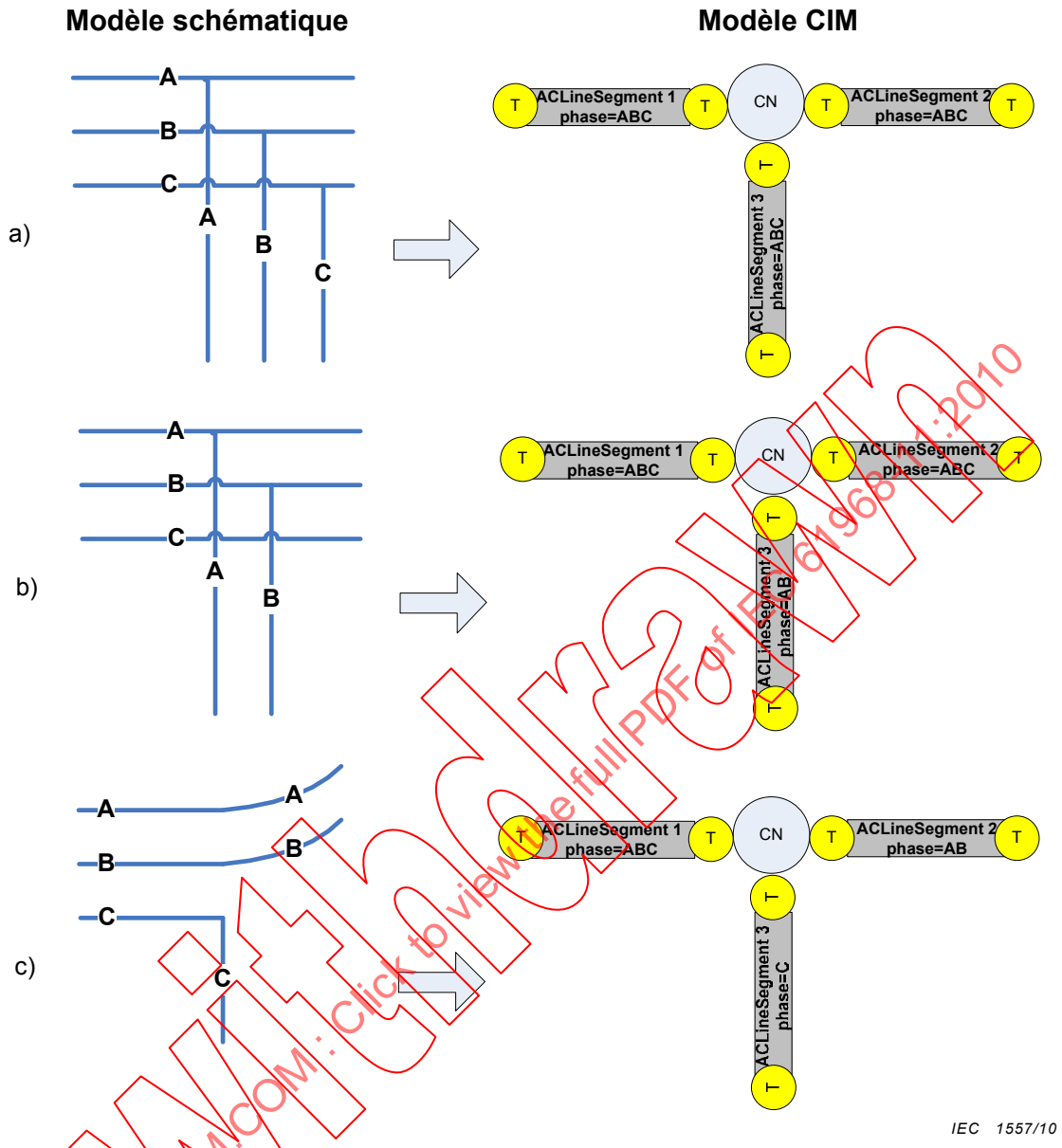


Figure 11 – Connectivité DCIM pour les dispositifs à deux Terminal (à deux bornes)

Une modélisation de connectivité similaire est à appliquer pour n'importe quels autres dispositifs à deux **Terminal** tels que shunts et commutateurs en série.

La Figure 12 illustre deux exemples pour la modélisation des charges: équilibrées et déséquilibrées.

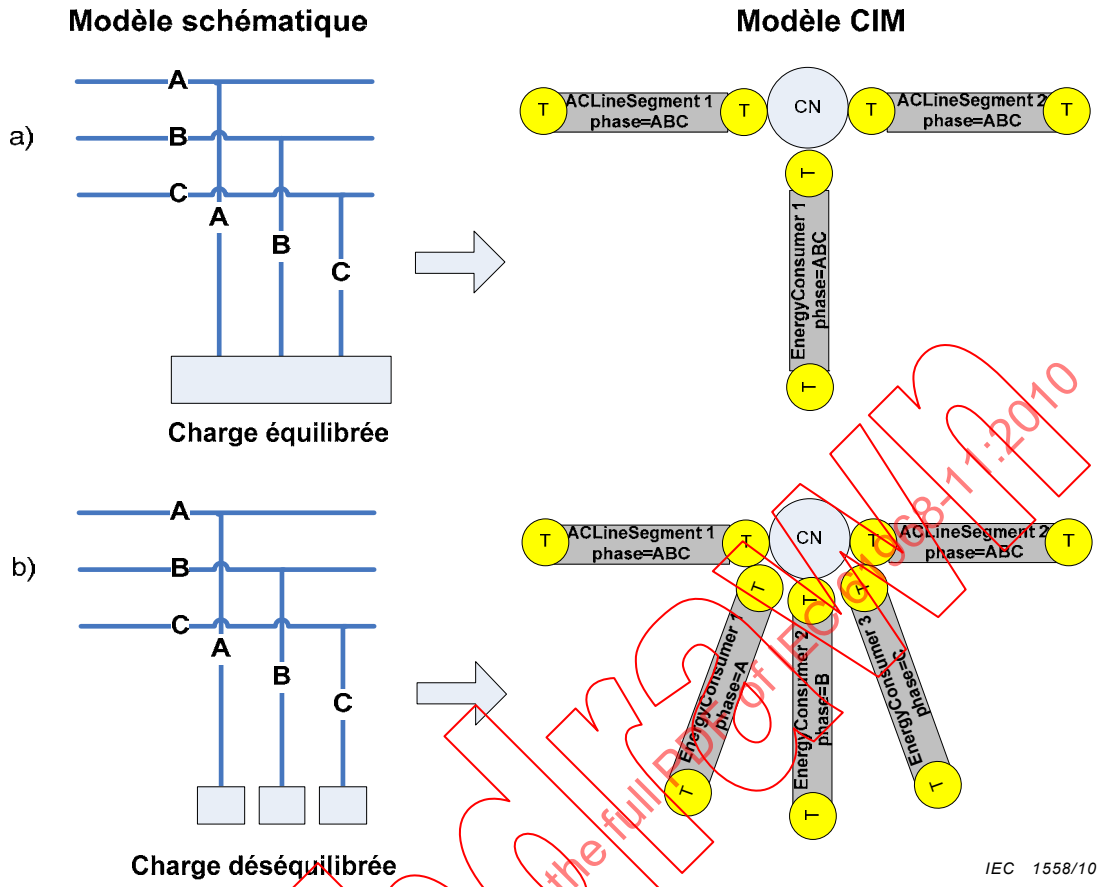


Figure 12 – Connectivité DCIM pour les dispositifs à un seul Terminal (à une seule borne)

Une modélisation de connectivité similaire est à appliquer pour n'importe quels autres dispositifs à un **Terminal** tels que shunts, générateurs et enroulements de transformateur.

NOTE La présente édition de la CEI 61968-11 ne prend pas en charge les commutateurs à états multiples (modélisés par **CompositeSwitch**) et ne fournit aucun concept de regroupement pour une batterie de commutateurs (dans laquelle les phases peuvent avoir des statuts différents), ou pour des charges (avec des caractéristiques de charge potentiellement différentes).

4.4.7 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. La représentation physique peut aussi être essentielle dans la dérivation des attributs pour la représentation électrique, même si aucune classe explicite **Asset** n'est utilisée. Le paragraphe 4.4.4 explique comment le modèle de biens peut aider à calculer la caractéristique électrique d'une ligne de distribution. La relation entre les deux aspects fournit aussi des informations cruciales pour la planification des extensions, la gestion des travaux et la gestion des interruptions de service.

La Figure 13 illustre comment les classes **Asset** et **PowerSystemResource** sont liées dans le CIM pour les besoins d'un certain nombre de scénarios d'échange de données.

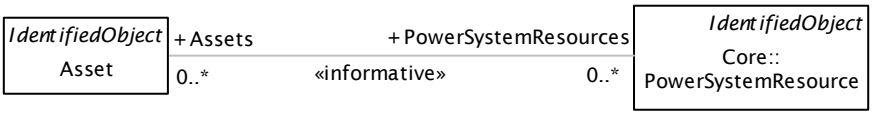


Figure 13 – Biens du DCIM et relation aux ressources du système d'énergie

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 caractéristiques assignées, statut d'alimentation en énergie) sont toujours modélisés par le truchement de **PowerSystemResource** et de ses sous-classes, alors que les aspects physiques (tels que caractéristiques assignées dans les feuilles de données, dimensions, cycle de vie des biens) sont toujours modélisés par le truchement de la classe **Asset** et de ses classes connexes.

NOTE La présente édition de la CEI 61968-11 contient les classes **Asset** et **AssetModel** car elles sont utilisées dans le contexte du mesurage (CEI 61968-9), mais pas dans le contexte de l'échange de modèles de réseau de distribution (CEI 61968-13). L'ensemble limité de classes relatives aux biens dans la présente édition contient les classes qui prennent en charge la modélisation des conducteurs et transformateurs seulement. En outre, la relation **Asset**-**PowerSystemResource** n'est pas utilisée, donc elle est étiquetée comme informative. Il convient que la prochaine édition de la CEI 61968-11 contienne un modèle de biens plus complet pour les besoins des échanges de modèles de réseau de distribution.

4.4.8 Emplacements et représentations graphiques

Pour permettre de localiser les différentes entités dans l'espace, le DCIM fournit la classe **Location**. Les entités peuvent être localisées par leurs différentes sortes d'adresses ainsi que par les coordonnées de position, **PositionPoint** dans un **CoordinateSystem** donné.

La classe **Location** est vide de tout échange de données graphiques, bien que ses **PositionPoint** puissent être utilisés pour échanger des coordonnées qui peuvent être utilisées par des applications graphiques.

La Figure 14 montre comment la classe **Location** peut être utilisée en relation avec des **Asset et PowerSystemResource**.

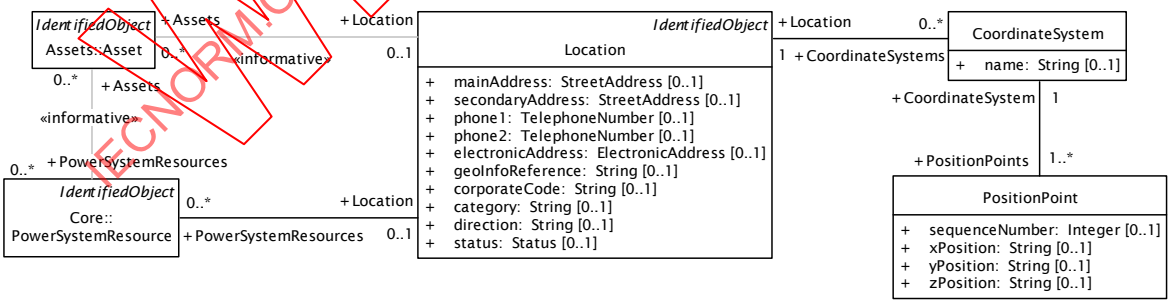


Figure 14 – Emplacements des biens et des ressources du système énergétique du modèle DCIM

NOTE Le support pour des échanges de schémas unifilaires, utilisés dans les opérations de transmission et certains centres de commande de distribution, est prévu être pris en charge dans le CIM de base, CEI 61970-301 et peut être citée à partir d'une édition future de la présente CEI 61968-11. La prise en charge des échanges de données géospatiales ne relève pas du domaine d'application de la présente CEI 61968-11, car il existe des normes largement utilisées et adoptées qui peuvent aussi être appliquées aux réseaux électriques.

4.4.9 Metering

Le paquetage **Metering** (comptage ou mesurage) introduit des classes, montrées à la Figure 15, qui sont nécessaires à l'intégration par les entreprise des systèmes de comptage et les informations qu'elles échangent avec d'autres systèmes d'entreprises. Un modèle logiciel pour un compteur et fourni par **MeterAsset**, qui hérite de **EndDeviceAsset**. **EndDeviceAssets** peut détecter les **EndDeviceEvents** et en rendre compte, rapporter les **MeterReadings** et accepter les **EndDeviceControls**. Dans le modèle pour les relevés de lecture de compteur, chaque relevé de lecture a un type spécifique et peut être associé à un intervalle. La communication de **EndDeviceAsset** peut être facilitée par des adresses individuelles ou des adresses de groupes. Les **EndDeviceAssets** sont déployés en des **ServiceDeliveryPoints**, fournissant les moyens d'assurer la relation à des emplacements et des clients. Une classe **MeterServiceWork** est introduite pour faciliter les travaux liés à l'installation, la maintenance et le remplacement de compteurs.

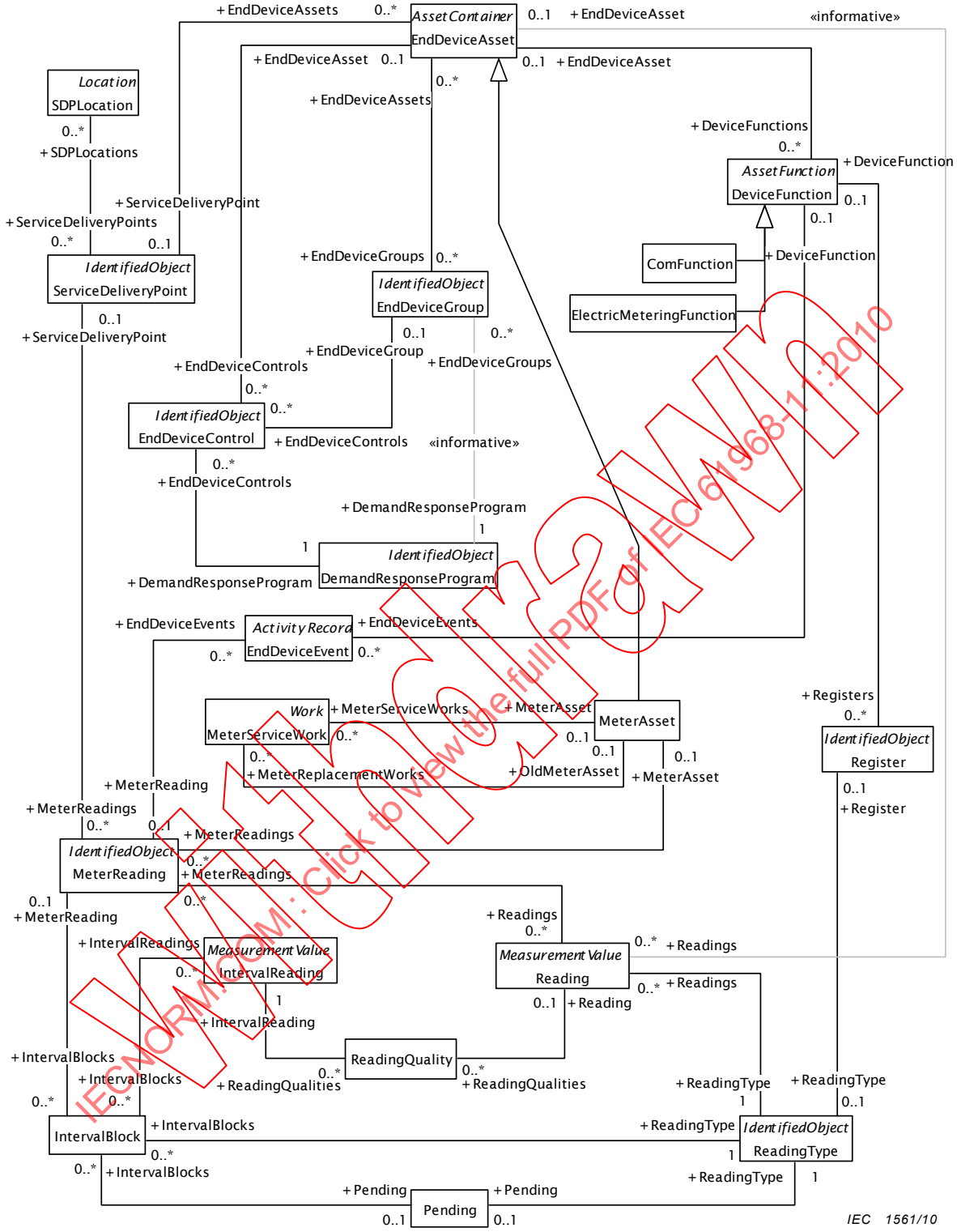


Figure 15 – Modèle de mesurage du DCIM

4.4.10 PaymentMetering

4.4.10.1 En transaction

Un système de mesurage de paiement 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'information de la clientèle. Un exemple type de la réalisation d'un tel plan d'enregistrement d'informations utilisant certaines des classes du CIM présentes dans le paquetage **PaymentMetering** est montré à la Figure 16.

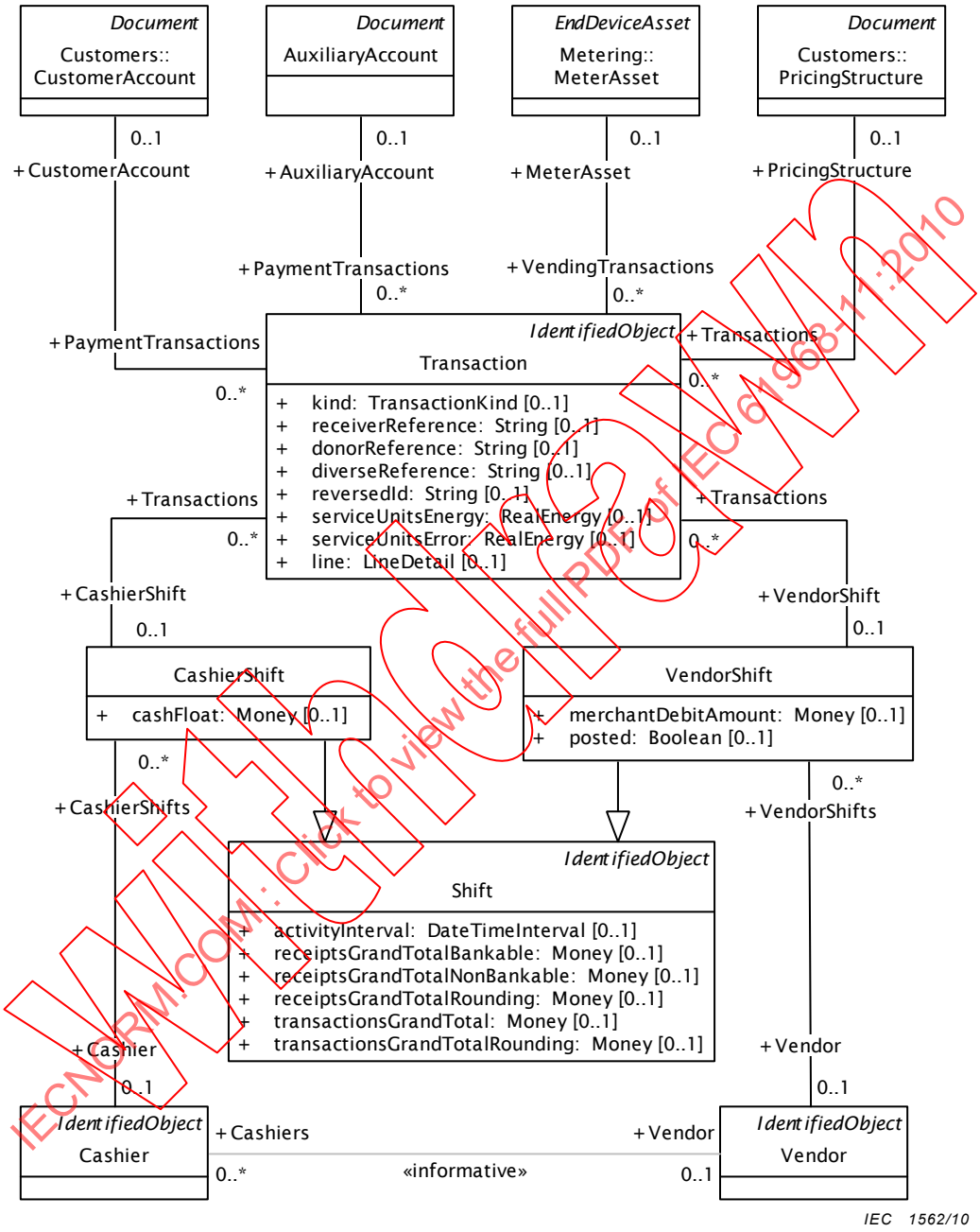


Figure 16 – Modèle de transaction du DCIM

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.

4.4.10.2 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 17.

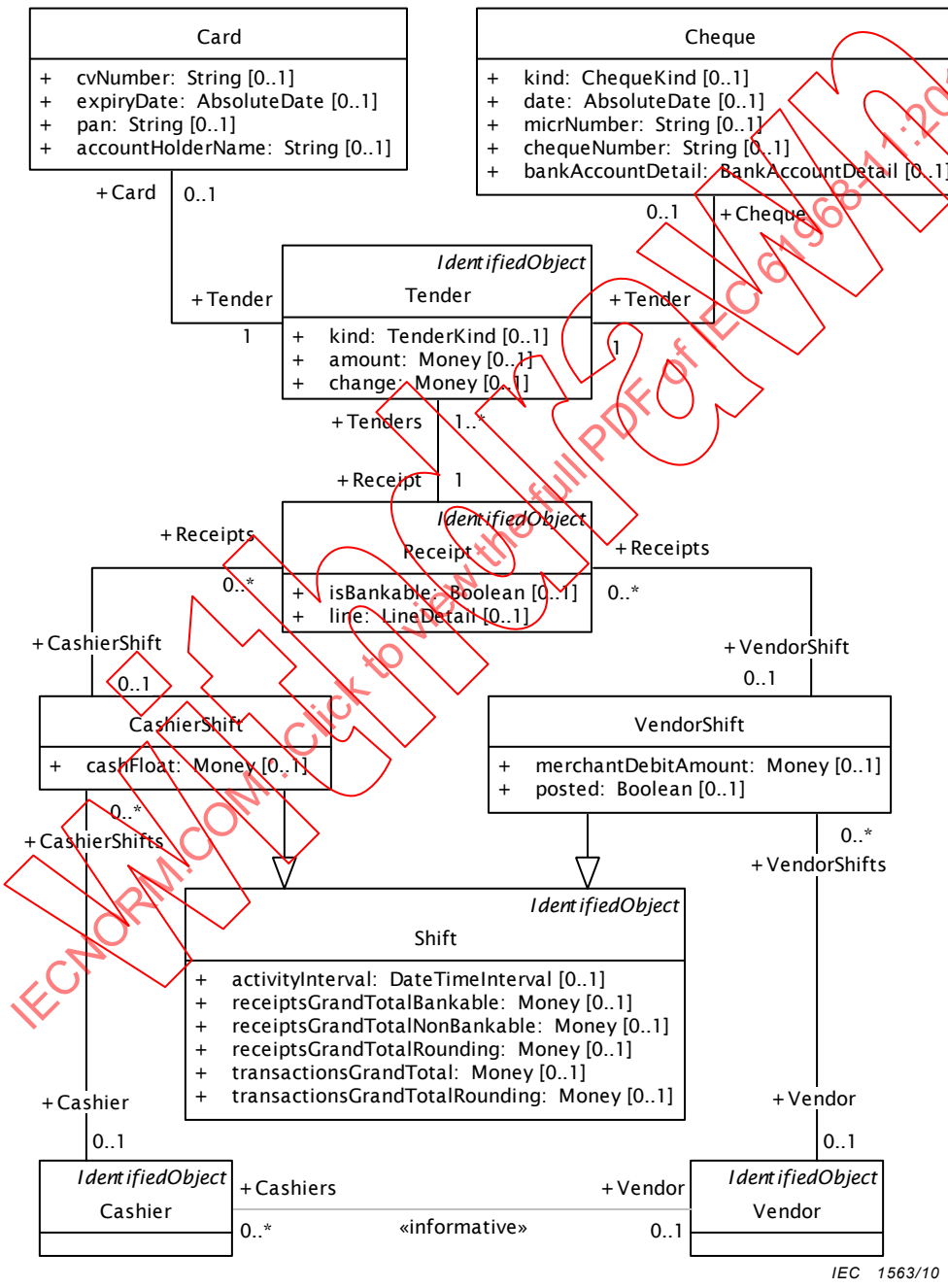


Figure 17 – Modèle d'acquittement du DCIM

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.

4.4.10.3 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 18.

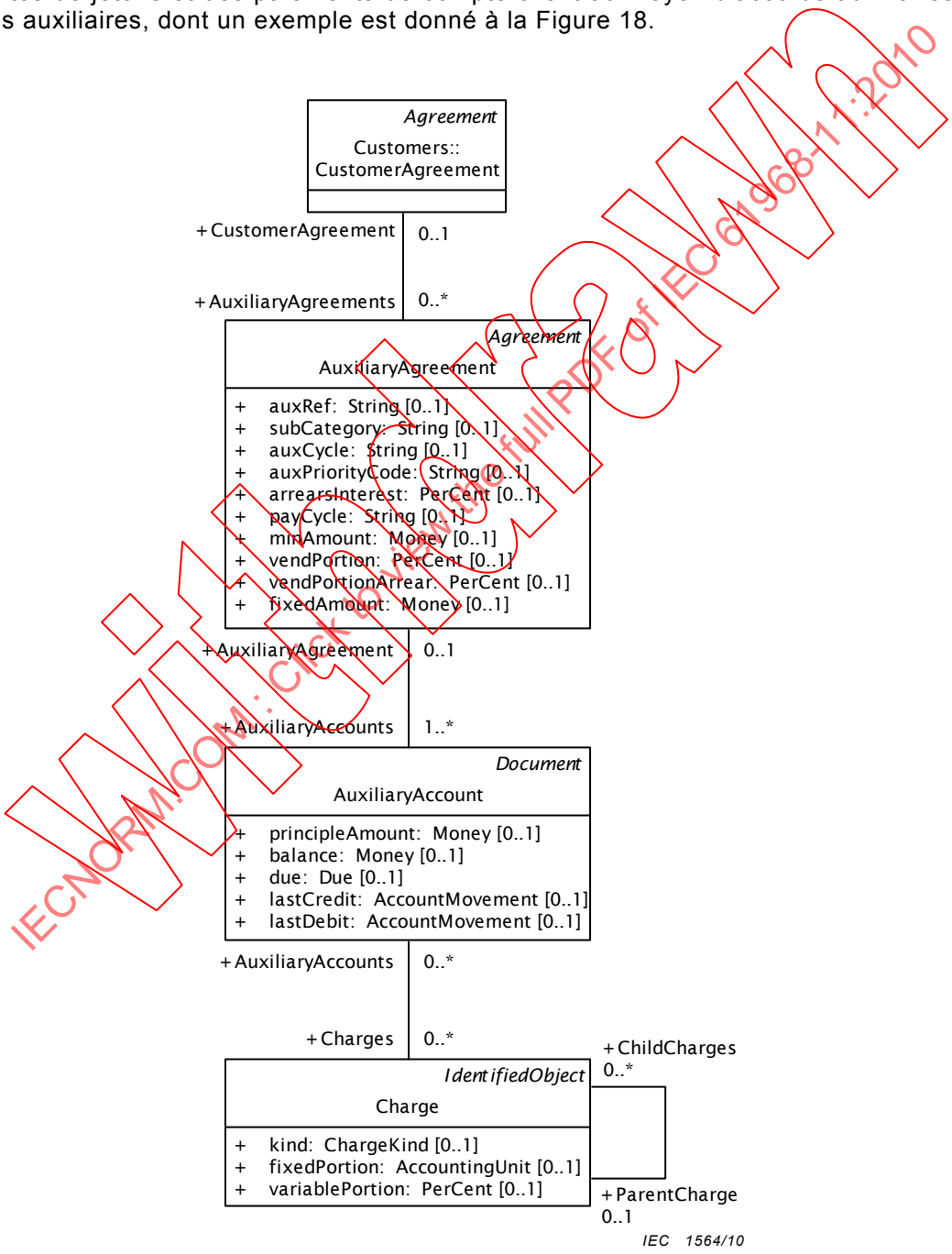


Figure 18 – Modèle d'accord auxiliaire du DCIM

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 perçues du compte **CustomerAccount** conformément aux règles établies dans **AuxiliaryAgreement** et prend en compte les coûts fixes, coûts variables et coûts en pourcentage.

4.4.10.4 Tarification et structures de barèmes

Les structures de tarification peuvent contenir des barèmes, dont la structure et le fonctionnement sont souvent assez complexes. La plupart des barèmes imposent des taxes qui sont basées sur la durée ou basées sur la consommation et, dans les deux cas, basées sur l'intervalle. Un modèle pour réaliser ces structures de barèmes complexes est montré à la Figure 19.

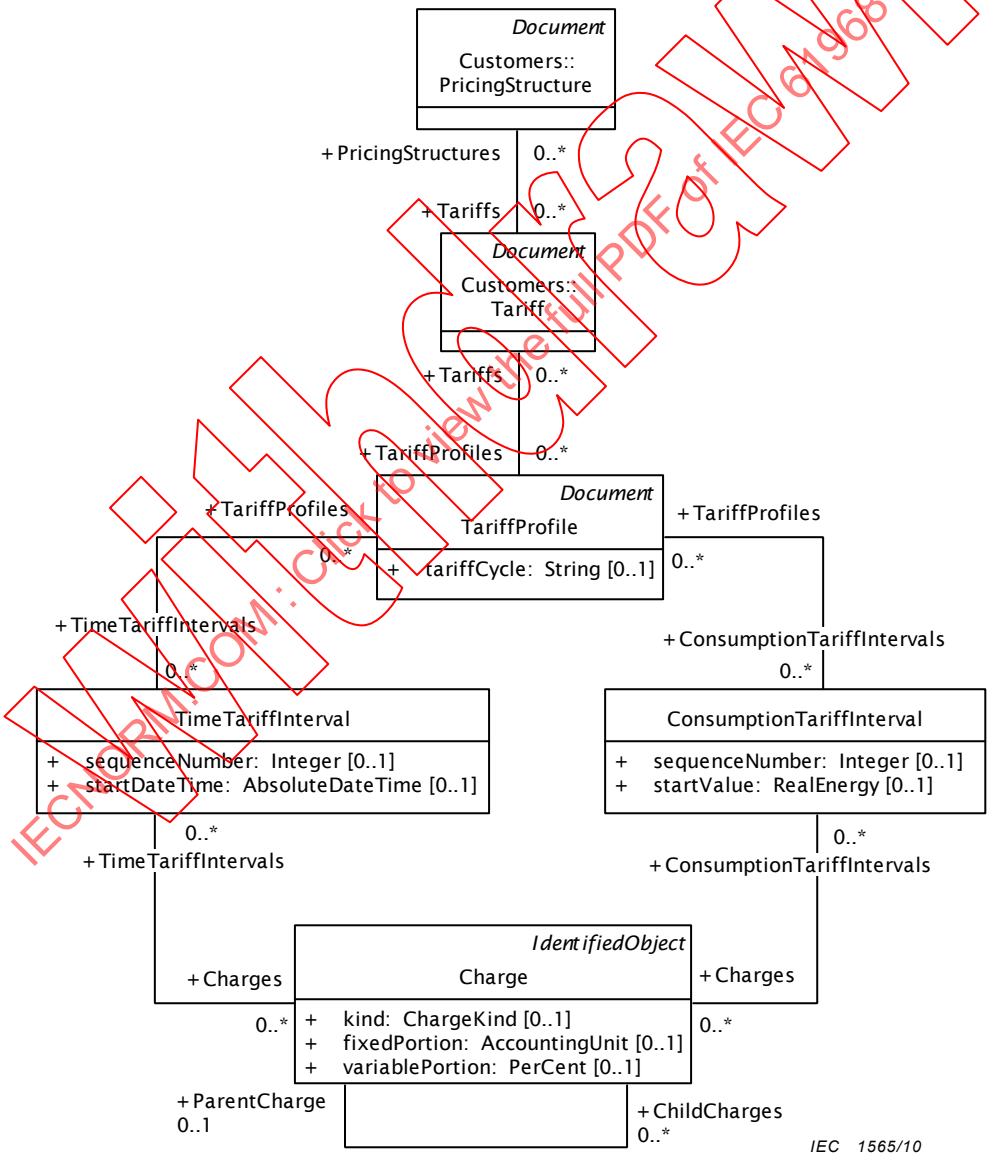


Figure 19 – Modèle de structure de tarification du DCIM

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 barème 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 barème à tranches ou un barème 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 taxes basées sur le temps et basées sur la consommation.

4.5 Autre

Les Paragraphes 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. Les points suivants ont changé depuis la dernière édition 1.0 publiée en 2003:

- Désormais, au lieu de Rational Rose, c'est Enterprise Architect qui est utilisé pour maintenir l'UML qui définit le DCIM. Le fichier Enterprise Architect (*.eap) le plus courant fournit la documentation DCIM la plus courante.
- CIMTool fournit une méthode permettant d'étendre le DCIM et de créer des profils, mais d'autres méthodes peuvent aussi être utilisées.
- La classe **Naming** est maintenant la classe **IdentifiedObject**. Les essais d'interopérabilité ont prouvé la nécessité de maintenir des identificateurs d'objets uniques et persistants à travers les domaines des modèles. Dans la pratique, un moyen pour ce faire serait l'utilisation d'identificateurs uniques universels (UUID, universally unique identifiers) pour l'attribut **mRID** de **IdentifiedObject**. Cependant, cela n'est pas une exigence.
- Les versions plus anciennes de l'UML CIM utilisent trois stéréotypes UML normalisés:Primitive, enumeration et Datatype. Ce dernier est toutefois utilisé avec une sémantique CIM spécifique (pas la sémantique UML normalisée): pour un triplet d'attributs {value, unit, multiplier}, qui implique un mappage personnalisé à des artéfacts de sérialisation (RDFS, OWL, XSD). Par conséquent, un nouveau stéréotype UML personnalisé, Compound, a été introduit pour "compléter" la sémantique de dataType de la norme UML: groupe de valeurs sans identité. Les classes ayant ce stéréotype ne participent jamais à des relations (généralisation, association), mais sont simplement utilisées comme types pour les attributs.

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.

5.2 Contexte

Le CIM est fractionné en sous-Paquetages. Les classes au sein des Paquetages sont répertoriées alphabétiquement. Les attributs natifs de la classe sont d'abord répertoriés, suivis des attributs hérités. Les associations natives sont d'abord répertoriées pour chaque classe, suivies des associations héritées. Les associations sont décrites selon le rôle de chaque classe participant à l'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 "note" contiendra le texte indiquant que les attributs sont hérités d'une classe spécifique. La colonne "note" 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 le tableau d'attributs, dans certains cas, un attribut est une constante, auquel cas l'expression «(const)» est ajouté 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'association

[mult à partir de]	[mult vers] nom	type	description
[0..*]	[0..*] OperatedBy_Companies	Company	hérité de: PowerSystemResource
[1]	[0..*] Contains_Measurements	Measurement	hérité de: PowerSystemResource
[1]	[0..*] OperatingShare	OperatingShare	hérité de: PowerSystemResource
[1..*]	[0..1] ModelingAuthoritySet	ModelingAuthoritySet	hérité de: IdentifiedObject

Dans le tableau extrémités d'association, 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 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 enumeration (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.

Tableau 4 – Documentation des Enums

libellé	description
steel	
lead	
lock	
other	

6 Architecture des paquetages (normative)

6.1 Généralités

La présente spécification est générée automatiquement à partir du fichier du modèle CIM iec61970cim14v13_iec61968cim10v31_combined.eap.

6.2 Paquetage supérieur IEC61968

Les sous-paquetages CEI 61968 du CIM sont développés, normalisés et maintenus par le groupe de travail 14 du comité d'études 57 de la CEI: Système d'interfaces pour la Gestion de la Distribution (SIDM, WG14).

A l'heure actuelle, les parties normatives du modèle viennent à l'appui des besoins d'échange d'informations définis dans la CEI 61968-9 et dans la CEI 61968-13.

La Figure 20 montre le diagramme de paquetages Main (Représentation de base).

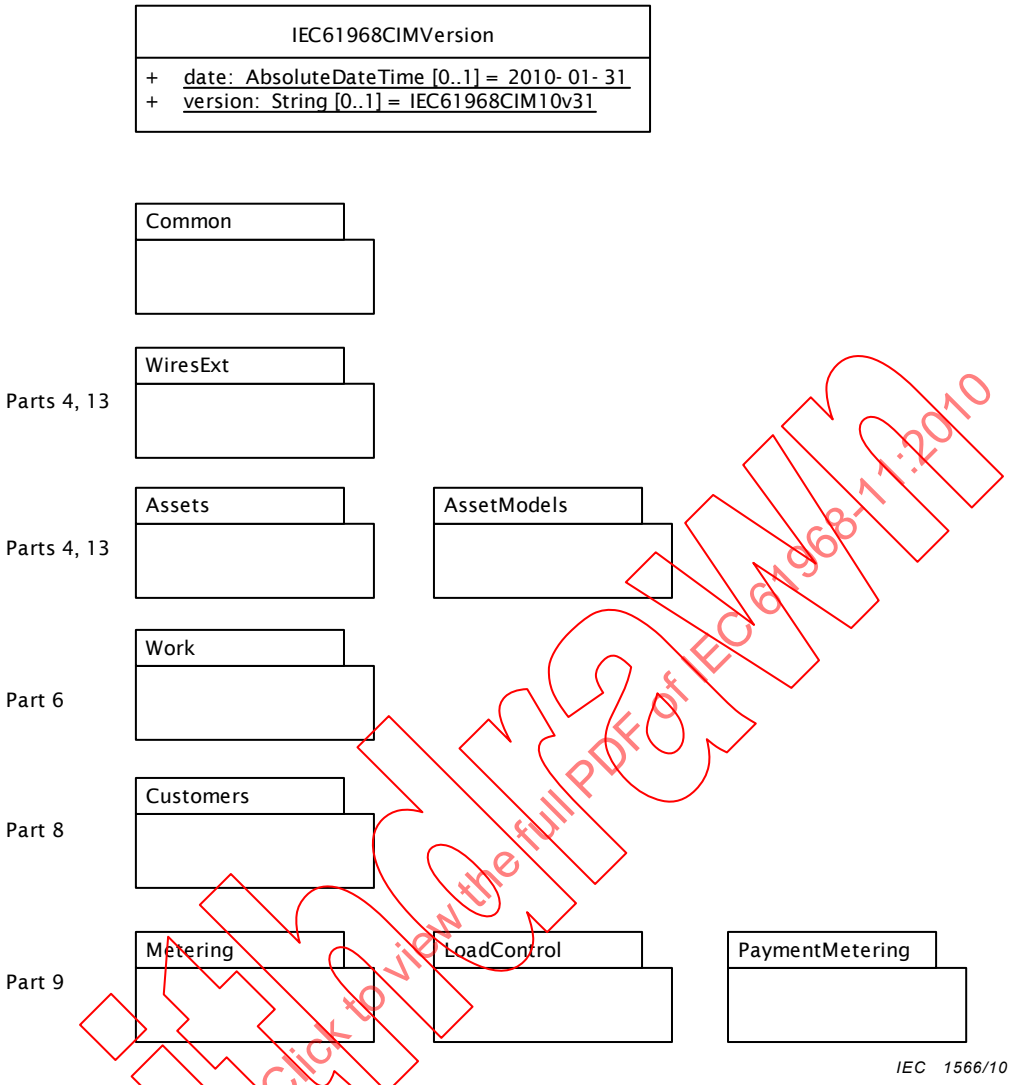


Figure 20 – Diagramme des paquetages IEC61968::Main

Ce diagramme montre la version et le contenu normatif des extensions du CIM pour la distribution.

L'annotation sur la gauche du diagramme indique la partie de la CEI 61968 qui pilote principalement les exigences relatives à la modélisation pour le(s) paquetage(s) respectif(s).

La Figure 21 montre le diagramme de paquetages Dependencies (Dépendances).

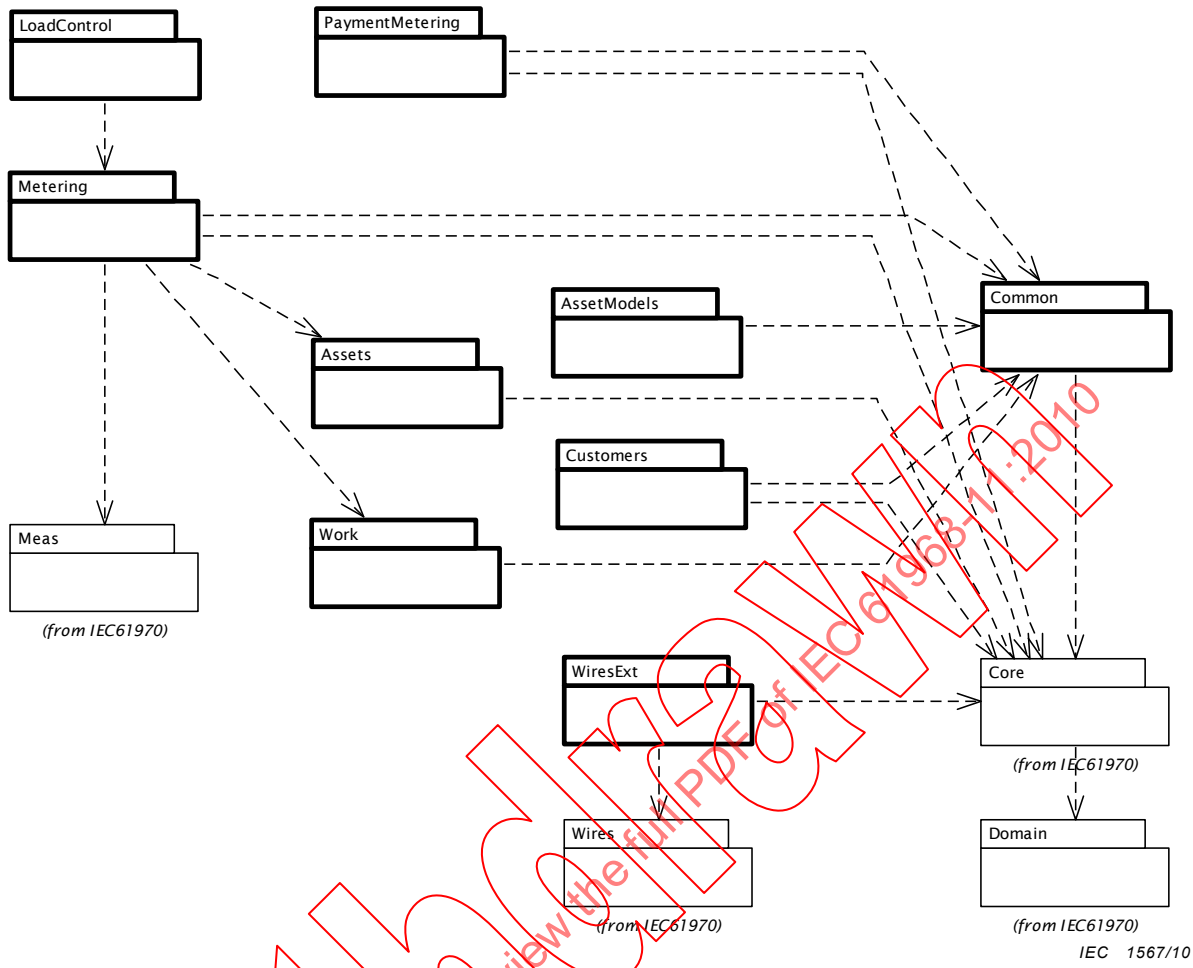


Figure 21 – Diagramme des paquetages IEC61968::Dependencies

Ce diagramme montre en gras le contenu de ce paquetage, ainsi que les dépendances basées sur l'héritage uniquement.

La Figure 22 montre le diagramme de paquetages StdCIM (CIM normalisé).

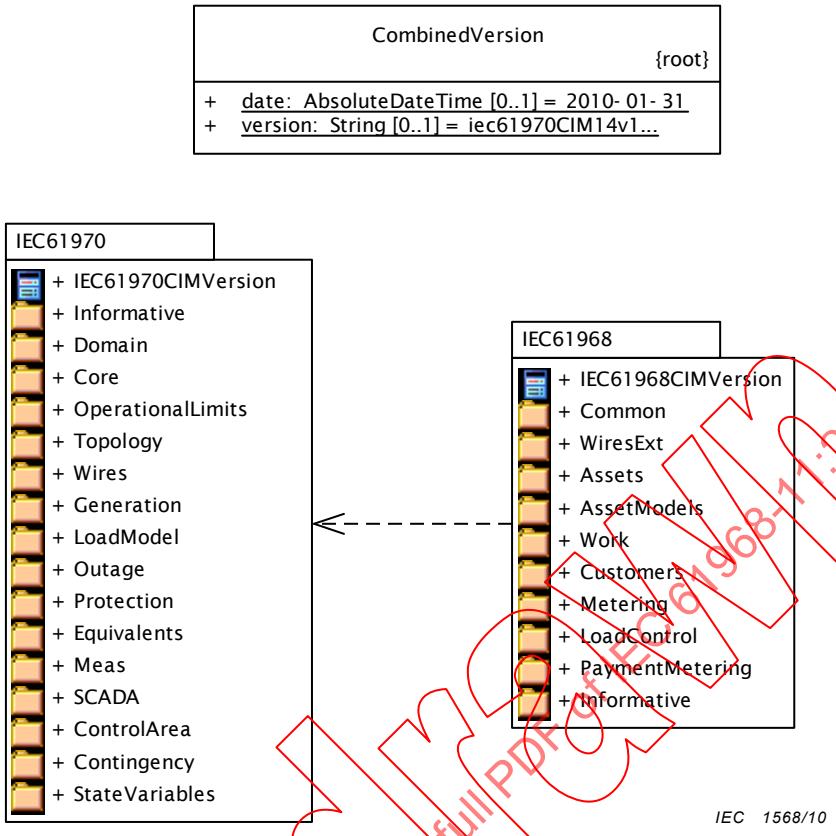


Figure 22 – Diagramme des paquetages IEC61968::StdCIM

Ce diagramme montre la version et le contenu normatif du modèle CIM actuellement défini du CE 57.

La Figure 23 montre le diagramme logique DCIMKeyClasses.

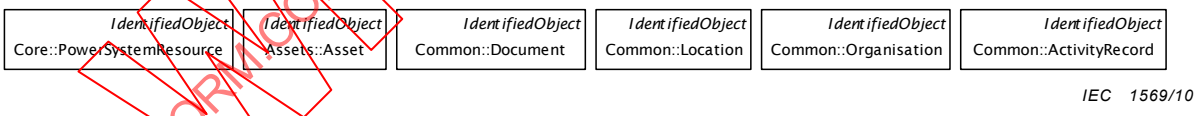


Figure 23 – Diagramme logique IEC61968::DCIMKeyClasses

Le diagramme montre les classes-clés dans le DCIM.

6.2.1 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=2010-01-31 (const)	AbsoluteDateTime	La forme est AAAA-MM-JJ par exemple le 5 janvier 2009 est 2009-01-05.
version=IEC61968CIM10v31 (const)	String	La forme est IEC61968CIMXXvYY où XX est la version principale du paquetage CIM et YY la version secondaire. Par exemple IEC61968CIM10v17.

6.2.2 Paquetage Common

6.2.2.1 Généralités

Ce paquetage contient les classes d'informations qui prennent en charge la gestion de la distribution en général.

La Figure 24 montre le diagramme logique CommonInheritance.

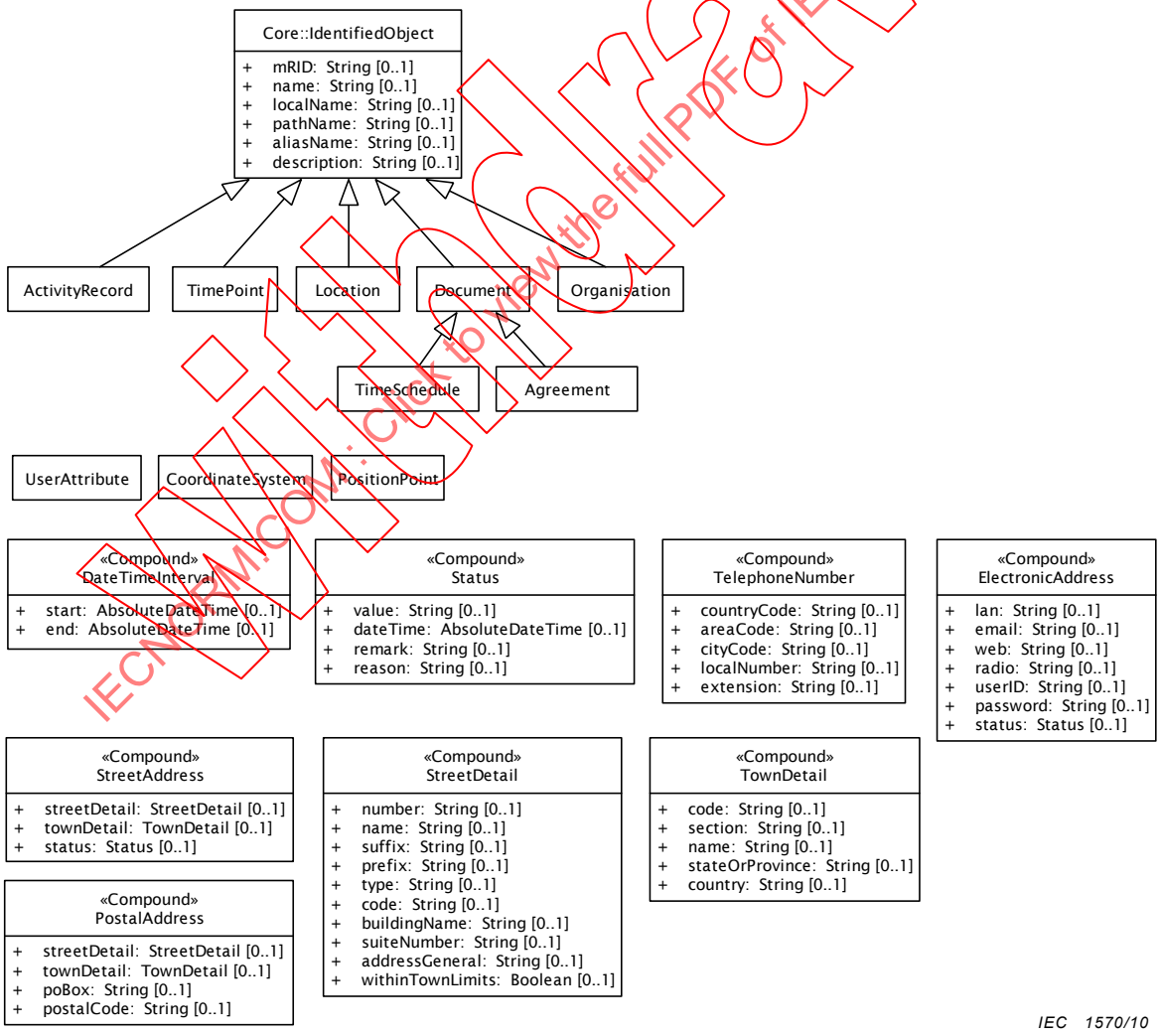


Figure 24 – Diagramme logique Common::CommonInheritance

Ce diagramme montre la hiérarchie d'héritage pour les classes normatives issues de ce paquetage, ainsi que les énumérations et les types compound.

La Figure 25 montre le diagramme logique CommonOverview.

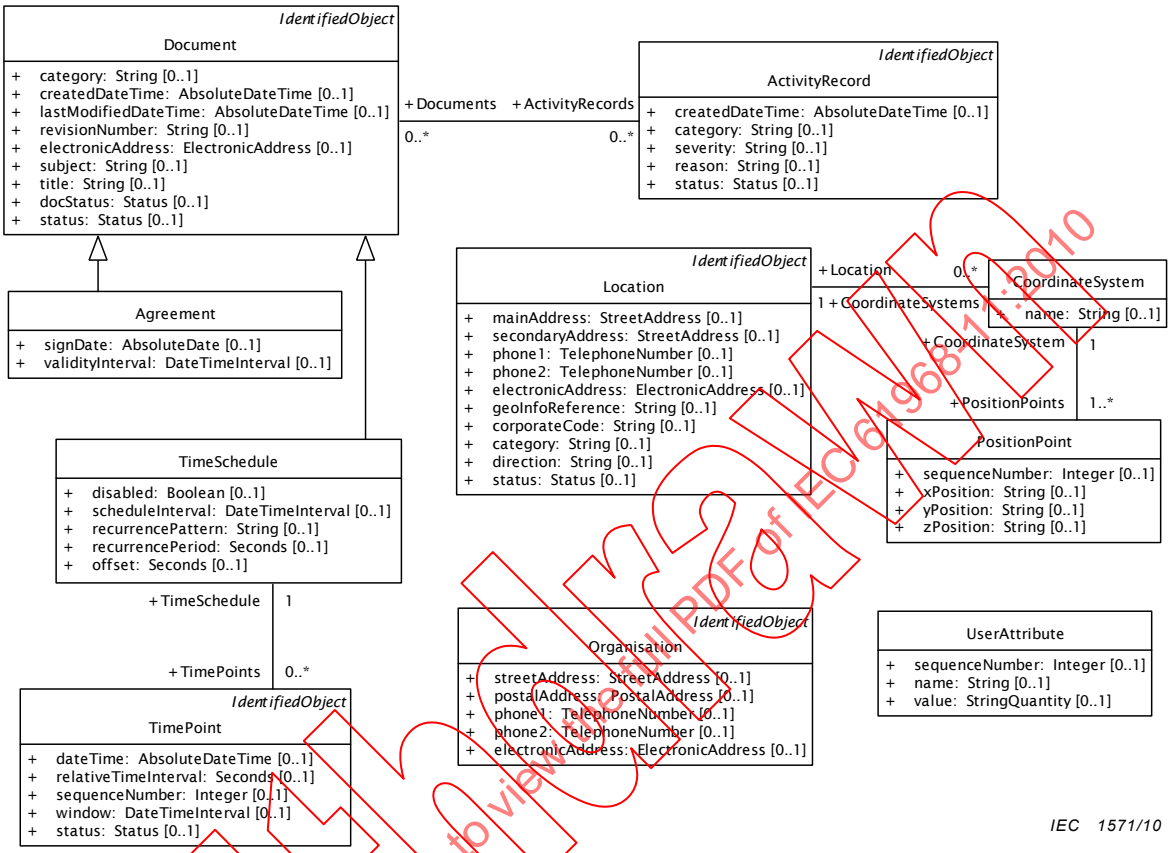


Figure 25 – Diagramme logique Common::CommonOverview

Ce diagramme montre les classes normatives issues de ce paquetage.

La Figure 26 montre le diagramme logique DCIMLocations.

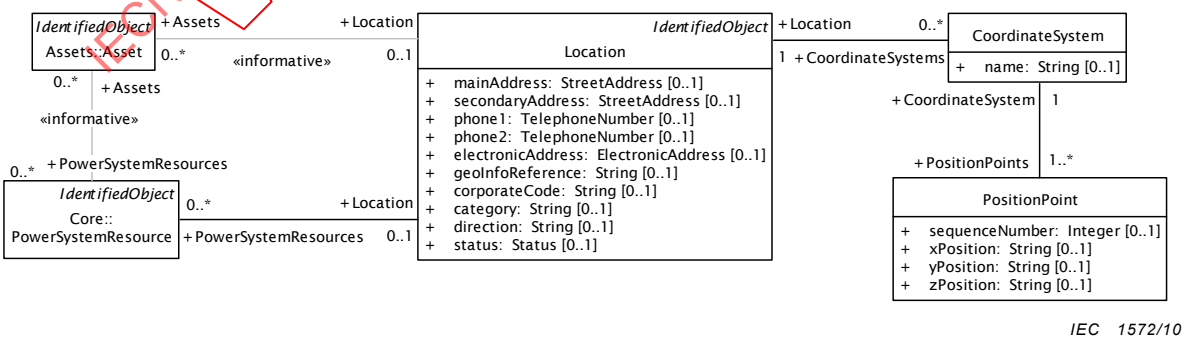


Figure 26 – Diagramme logique Common::DCIMLocations

Ce diagramme montre les classes utilisées pour modéliser les emplacements de ressources de système de puissance et des biens dans le DCIM.

6.2.2.2 Compound DateTimeInterval

Intervalle de date et de temps.

Le Tableau 6 montre tous les attributs de DateTimeInterval.

Tableau 6 – Attributs de Common::DateTimeInterval

nom	type	description
start	AbsoluteDateTime	Date et heure auxquelles cet intervalle a commencé.
end	AbsoluteDateTime	Date et heure auxquelles cet intervalle s'est terminé.

6.2.2.3 Compound Status

Informations relatives au statut courant pertinentes pour une entité.

Le Tableau 7 montre tous les attributs de Status.

Tableau 7 – 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	AbsoluteDateTime	Date et heure pour lesquelles «value» de status s'applique.
remark	String	Informations pertinentes relatives à la «value» (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.

6.2.2.4 Compound PostalAddress

Informations d'usage général relatives à l'adresse postale.

Le Tableau 8 montre tous les attributs de PostalAddress.

Tableau 8 – 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.

6.2.2.5 Compound StreetAddress

Informations d'usage général relatives à l'adresse de la rue.

Le Tableau 9 montre tous les attributs de StreetAddress.

Tableau 9 – 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.

6.2.2.6 Compound StreetDetail

Détails relatifs à la rue, dans le contexte de l'adresse.

Le Tableau 10 montre tous les attributs de StreetDetail.

Tableau 10 – 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.
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	Information complémentaires relatives à l'adresse, par exemple une case postale.
withinTownLimits	Boolean	True (vrai) si cette rue se situe dans les limites géographiques légales de la ville spécifiée (valeur par défaut).

6.2.2.7 Compound TownDetail

Détails relatives à la ville, dans le contexte de l'adresse.

Le Tableau 11 montre tous les attributs de TownDetail.

Tableau 11 – 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.

6.2.2.8 Compound ElectronicAddress

Informations relatives à l'adresse électronique.

Le Tableau 12 montre tous les attributs de ElectronicAddress.

Tableau 12 – Attributs de Common::ElectronicAddress

nom	type	description
lan	String	Adresse sur le réseau local.
email	String	Adresse de courriel (email)
web	String	Adresse web (Toile mondiale) .
radio	String	Adresse radio
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.
status	Status	Statut de l'adresse électronique.

6.2.2.9 Compound TelephoneNumber

Numéro de téléphone.

Le Tableau 13 montre tous les attributs de TelephoneNumber.

Tableau 13 – 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.

6.2.2.10 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 14 montre tous les attributs de ActivityRecord.

Tableau 14 – Attributs de Common::ActivityRecord

nom	type	description
createdDateTime	AbsoluteDateTime	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).
category	String	Catégorie d'événement entraînant cet enregistrement d'activité.
severity	String	Niveau de gravité 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 (abréviation)	String	hérité de: IdentifiedObject
description	String	hérité de: IdentifiedObject
localName	String	hérité de: IdentifiedObject
mRID	String	hérité de: IdentifiedObject
name	String	hérité de: IdentifiedObject
pathName	String	hérité de: IdentifiedObject

Le Tableau 15 montre toutes les extrémités d'association de ActivityRecord avec les autres classes.

Tableau 15 – Extrémités d'association 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..*]	[0..*] Documents	Document	Tous les documents pour lesquels cet enregistrement d'activité a été créé.

6.2.2.11 Agreement (Accord)

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 16 montre tous les attributs de Agreement.

Tableau 16 – Attributs de Common::Agreement

nom	type	description
signDate	AbsoluteDate	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).
category	String	hérité de: Document
createdDateTime	AbsoluteDateTime	hérité de: Document
lastModifiedDateTime	AbsoluteDateTime	hérité de: Document
revisionNumber	String	hérité de: Document
subject	String	hérité de: Document
title	String	hérité de: Document
docStatus	Status	hérité de: Document
status	Status	hérité de: Document
electronicAddress	ElectronicAddress	hérité de: Document
aliasName	String	hérité de: IdentifiedObject
description	String	hérité de: IdentifiedObject
localName	String	hérité de: IdentifiedObject
mRID	String	hérité de: IdentifiedObject
name	String	hérité de: IdentifiedObject
pathName	String	hérité de: IdentifiedObject

Le Tableau 17 montre toutes les extrémités d'association de Agreement avec les autres classes.

Tableau 17 – Extrémités d'association de Common::Agreement avec les autres classes

[mult à partir de]	[mult vers] nom	type	description
[0..*]	[0..*] ActivityRecords	ActivityRecord	hérité de: Document

6.2.2.12 CoordinateSystem

Système de référence de coordonnées

Le Tableau 18 montre tous les attributs de CoordinateSystem.

Tableau 18 – Attributs de Common::CoordinateSystem

nom	type	description
name	String	Nom de ce système de coordonnées.

Le Tableau 19 montre toutes les extrémités d'association de CoordinateSystem avec les autres classes.

Tableau 19 – Extrémités d'association de Common:: CoordinateSystem avec les autres classes

[mult à partir de]	[mult vers] nom	type	description
[1..1]	[1..*] PositionPoints	PositionPoint	Séquence des points de position exprimée dans ce système de coordonnées.
[0..*]	[1..1] Location	Location	Emplacement décrit par les points de position de ce système de coordonnées.

6.2.2.13 Document

Classe parente pour les différents groupements d'informations recueillies et 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 de puissance.

Le Tableau 20 montre tous les attributs de Document.

Tableau 20 – Attributs de Common::Document

nom	type	description
category	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	AbsoluteDateTime	Date et heure auxquelles ce document a été créé.
lastModifiedDateTime	AbsoluteDateTime	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.
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».
electronicAddress	ElectronicAddress	Adresse électronique.
aliasName	String	hérité de: IdentifiedObject
description	String	hérité de: IdentifiedObject
localName	String	hérité de: IdentifiedObject
mRID	String	hérité de: IdentifiedObject
name	String	hérité de: IdentifiedObject
pathName	String	hérité de: IdentifiedObject

Le Tableau 21 montre toutes les extrémités d'association de Document avec les autres classes.

Tableau 21 – Extrémités d'association de Common::Document avec les autres classes

[mult à partir de]	[mult vers] nom	type	description
[0..*]	[0..*] ActivityRecords	ActivityRecord	Tous les enregistrements d'activité créés pour ce document.

6.2.2.14 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 22 montre tous les attributs de Location.

Tableau 22 – Attributs de Common::Location

nom	type	description
category	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).
corporateCode	String	Code spécifique à l'entreprise de service public pour l'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».
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 «ServiceDeliveryPoint» peut être situé au deuxième étage d'un immeuble d'appartements.
geoInfoReference	String	(si applicable) Référence à une source d'informations géographiques, souvent extérieure à l'entreprise de service public.
status	Status	Statut de cet emplacement.
phone1	TelephoneNumber	Numéro de téléphone.
phone2	TelephoneNumber	Numéro de téléphone supplémentaire.
electronicAddress	ElectronicAddress	Adresse électronique.
aliasName	String	hérité de: IdentifiedObject
description	String	hérité de: IdentifiedObject
localName	String	hérité de: IdentifiedObject
mRID	String	hérité de: IdentifiedObject
name	String	hérité de: IdentifiedObject
pathName	String	hérité de: IdentifiedObject

Le Tableau 23 montre toutes les extrémités d'association de Location avec les autres classes.

Tableau 23 – Extrémités d'association 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 de puissance en cet emplacement.
[1..1]	[0..*] CoordinateSystems	CoordinateSystem	Tous les systèmes de coordonnées utilisés pour décrire les points de position de cet emplacement.

6.2.2.15 Organisation

Organisation susceptibles d'avoir des rôles d'entreprise de service public, d'entrepreneur, de fournisseur, de fabricant, de client, etc.

Le Tableau 24 montre tous les attributs de Organisation.

Tableau 24 – 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.
electronicAddress	ElectronicAddress	Adresse électronique.
aliasName	String	hérité de: IdentifiedObject
description	String	hérité de: IdentifiedObject
localName	String	hérité de: IdentifiedObject
mRID	String	hérité de: IdentifiedObject
name	String	hérité de: IdentifiedObject
pathName	String	hérité de: IdentifiedObject

6.2.2.16 PositionPoint

Jeu de coordonnées spatiales qui déterminent un point. Utiliser une seule instance de point de position pour décrire un emplacement orienté point. Utiliser une séquence de points de position pour décrire un objet orienté ligne (emplacement physique d'objets non orientés point 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 25 montre tous les attributs de PositionPoint.

Tableau 25 – 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 26 montre toutes les extrémités d'association de PositionPoint avec les autres classes.

Tableau 26 – Extrémités d'association de Common:: PositionPoint avec les autres classes

[mult à partir de]	[mult vers] nom	type	description
[1..*]	[1..1] CoordinateSystem	CoordinateSystem	Système de coordonnées dans lequel les coordonnées de ce point de position sont exprimées.

6.2.2.17 TimePoint

Instant dans une séquence d'instants relatifs à un TimeSchedule (à un calendrier général).

Le Tableau 27 montre tous les attributs de TimePoint.

Tableau 27 – Attributs de Common::TimePoint

nom	type	description
dateTime	AbsoluteDateTime	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
description	String	hérité de: IdentifiedObject
localName	String	hérité de: IdentifiedObject
mRID	String	hérité de: IdentifiedObject
name	String	hérité de: IdentifiedObject
pathName	String	hérité de: IdentifiedObject

Le Tableau 28 montre toutes les extrémités d'association de TimePoint avec les autres classes.

Tableau 28 – Extrémités d'association de Common::TimePoint avec les autres classes

[mult à partir de]	[mult vers] nom	type	description
[0..*]	[1..1] TimeSchedule	TimeSchedule	Calendrier général auquel cet instant appartient.

6.2.2.18 TimeSchedule

Description de tout ce qui change avec le temps. Le calendrier général est utilisé pour accomplir une fonction simplement valuée du temps. Utiliser l'attribut hérité «category» 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 29 montre tous les attributs de TimeSchedule.

Tableau 29 – Attributs de Common::TimeSchedule

nom	type	description
disabled	Boolean	True (vrai) si ce calendrier est désactivé.
scheduleInterval	DateTimeInterval	Intervalle de date et de temps du calendrier.
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 à 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 de lecture 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.
category	String	hérité de: Document
createdDateTime	AbsoluteDateTime	hérité de: Document
lastModifiedDateTime	AbsoluteDateTime	hérité de: Document
revisionNumber	String	hérité de: Document
subject	String	hérité de: Document
title	String	hérité de: Document
docStatus	Status	hérité de: Document
status	Status	hérité de: Document
electronicAddress	ElectronicAddress	hérité de: Document
aliasName	String	hérité de: IdentifiedObject
description	String	hérité de: IdentifiedObject

nom	type	description
localName	String	hérité de: IdentifiedObject
mRID	String	hérité de: IdentifiedObject
name	String	hérité de: IdentifiedObject
pathName	String	hérité de: IdentifiedObject

Le Tableau 30 montre toutes les extrémités d'association de TimeSchedule avec les autres classes.

Tableau 30 – Extrémités d'association 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 calendrier général.
[0..*]	[0..*] ActivityRecords	ActivityRecord	hérité de: Document

6.2.2.19 UserAttribute

Classe générique de paires nom-valeur avec numéro de séquence et unités facultatifs pour la valeur; Peut être utilisé pour modéliser des parties de l'échange d'informations lorsque des types concrets ne sont pas connus à l'avance.

Le Tableau 31 montre tous les attributs de UserAttribute.

Tableau 31 – 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 32 montre toutes les extrémités d'association de UserAttribute avec les autres classes.

Tableau 32 – Extrémités d'association 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.

6.2.3 Paquetage WiresExt

6.2.3.1 Généralités

Ce paquetage contient les classes d'informations qui étendent le paquetage IEC61970::Wires avec des ressources du système de puissance requises pour la modélisation du réseau de distribution, y compris les réseaux déséquilibrés.

La Figure 27 montre le diagramme logique WiresExtInheritance.

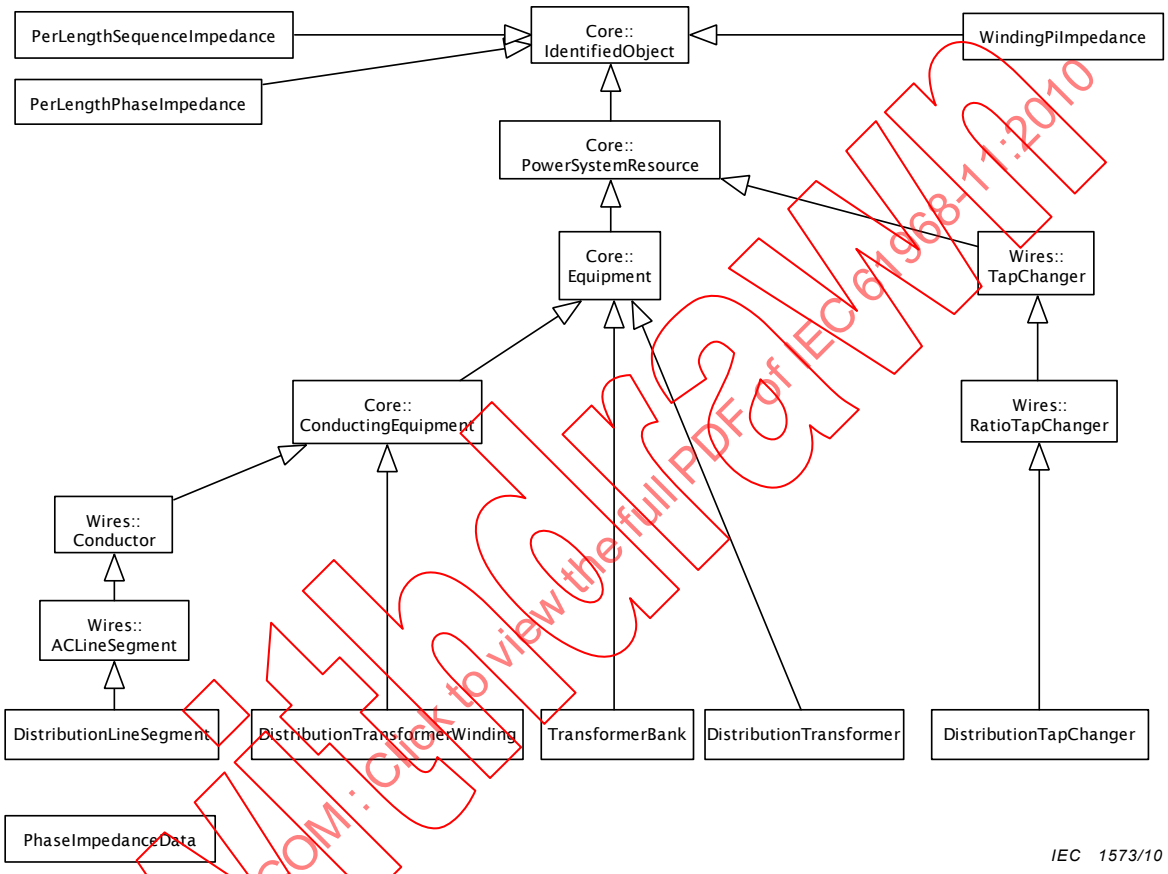


Figure 27 – Diagramme logique WiresExt::WiresExtInheritance

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 28 montre le diagramme logique DCIMLoadModel.

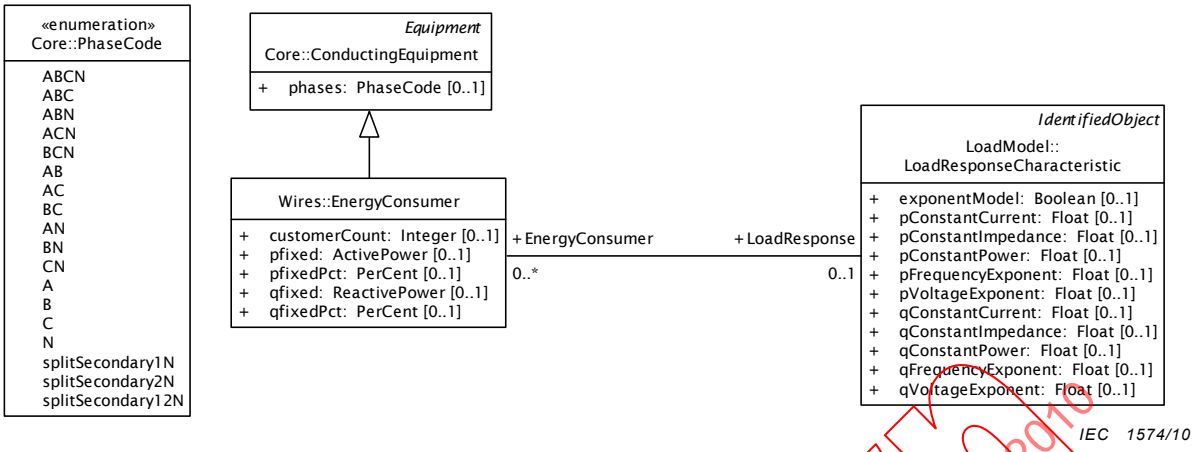


Figure 28 – Diagramme logique WiresExt::DCIMLoadModel

Ce diagramme montre les classes utilisées pour modéliser les charges dans le DCIM.

La Figure 29 montre le diagramme logique DCIMLineModel.

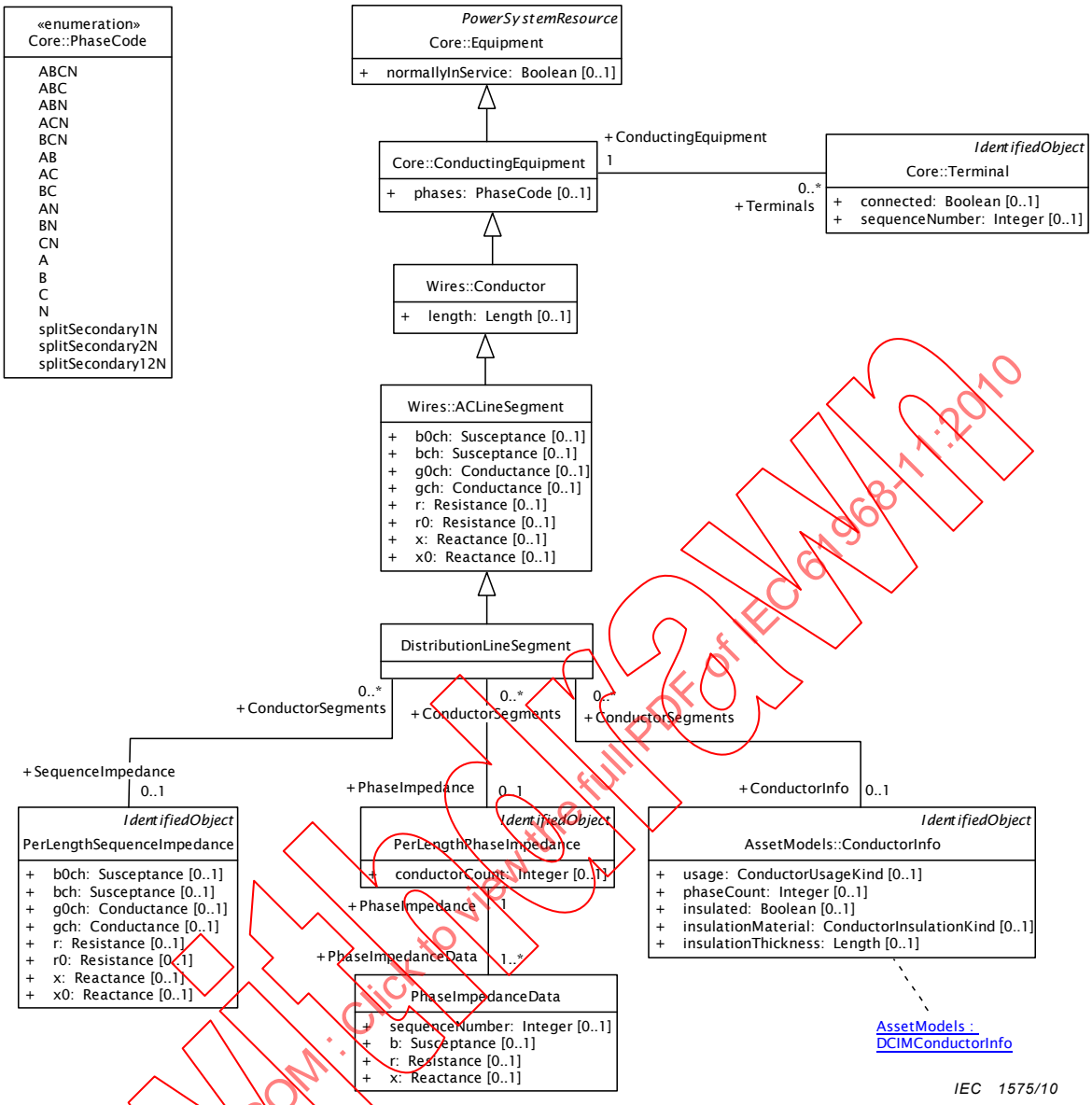


Figure 29 – Diagramme logique WiresExt::DCIMLineModel

Ce diagramme montre une partie des classes utilisées pour modéliser des lignes et des circuits dans le DCIM.

La Figure 30 montre le diagramme logique DCIMTransformerModel.

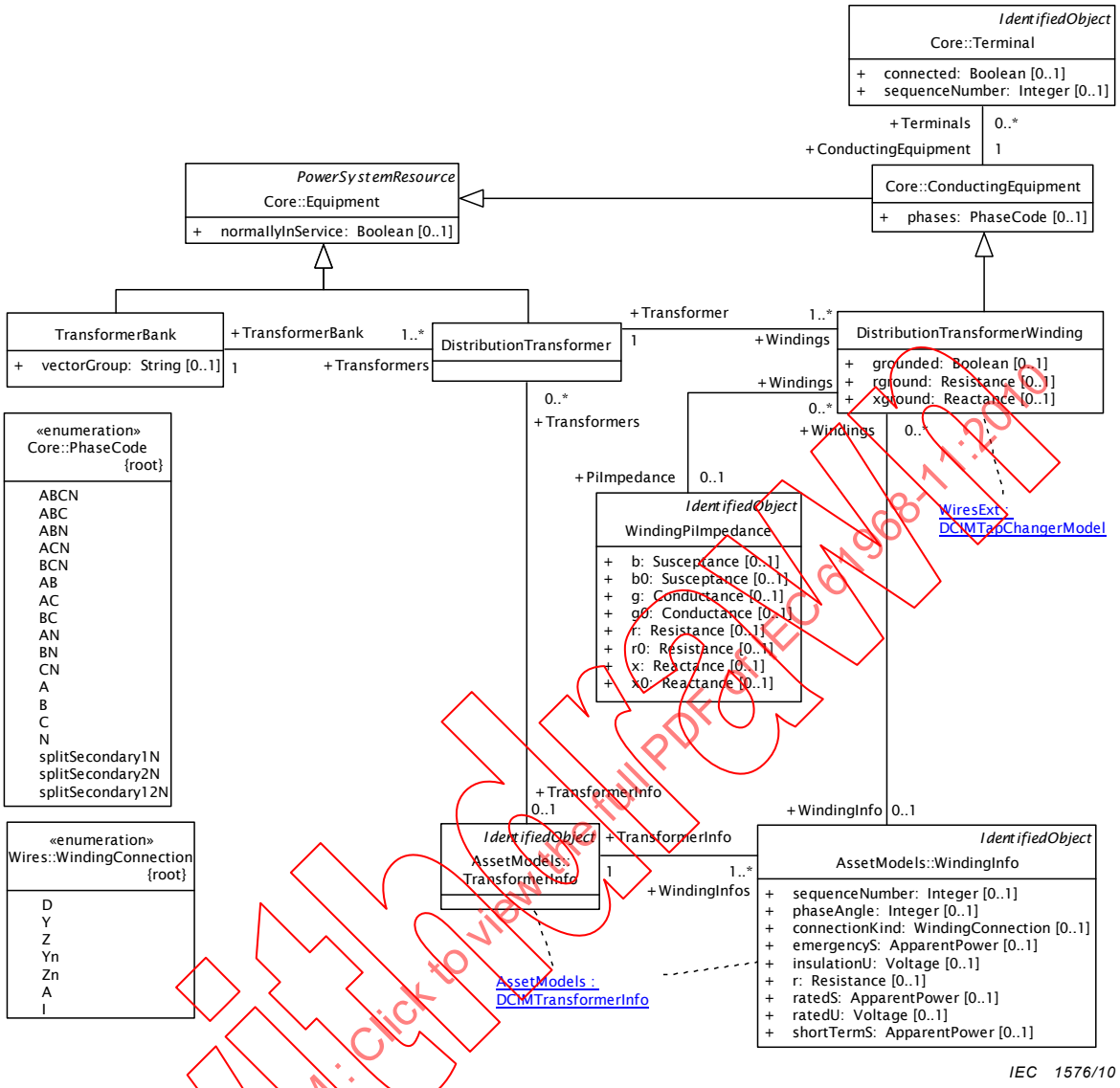


Figure 30 – Diagramme logique WiresExt::DCIMTransformerModel

Ce diagramme montre une partie des classes utilisées pour modéliser les transformateurs dans le DCIM.

La Figure 31 montre le diagramme logique DCIMTapChangerModel.

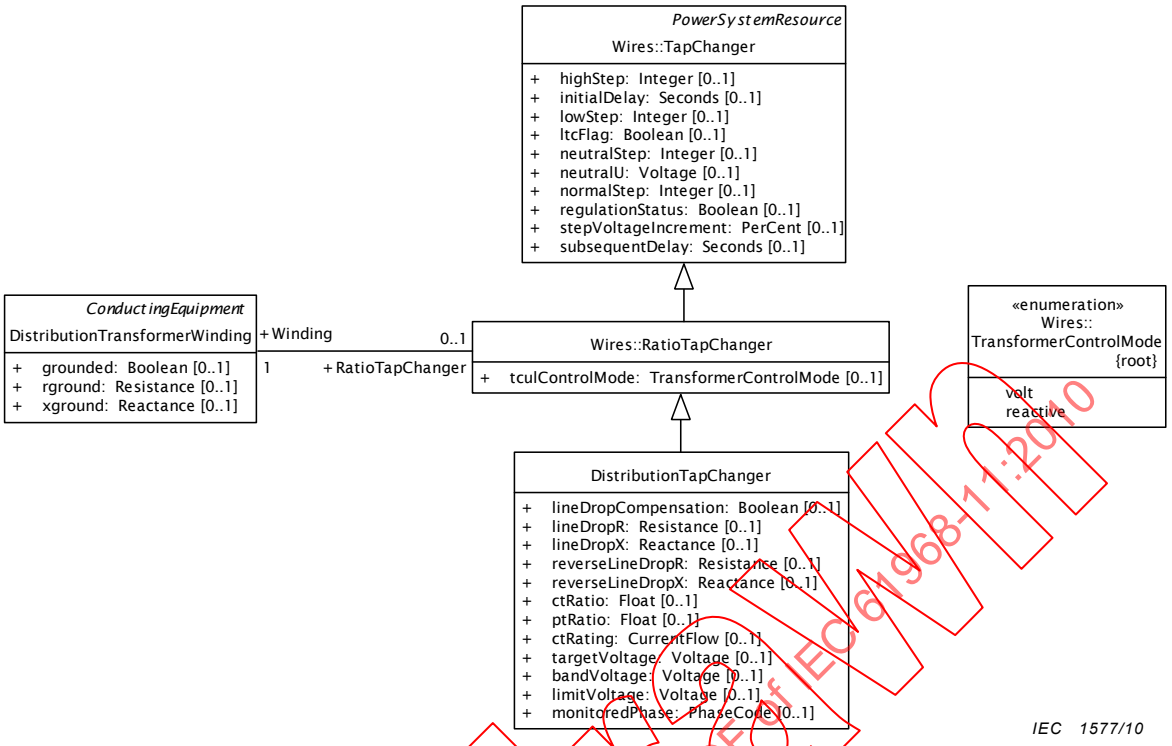


Figure 31 – Diagramme logique WiresExt::DCIMTapChangerModel

Ce diagramme montre les classes utilisées pour modéliser les changeurs de prises dans le DCIM.

6.2.3.2 DistributionLineSegment

Étend ACLineSegment avec des références à une bibliothèque de types normalisés à partir desquels les paramètres électriques peuvent être calculés, comme suit:

– calculer les paramètres électriques à partir des données relatives aux biens, en utilisant le ConductorInfo, puis multiplier les valeurs par Conductor.length pour produire un modèle de matrice.

– calculer les paramètres électriques déséquilibrés à partir PerLengthPhaseImpedance associée, puis multipliés par Conductor.length pour produire un modèle de matrice.

– calculer les paramètres électriques transposés à partir de PerLengthSequenceImpedance associée, puis multiplier par Conductor.length pour produire un modèle de séquence.

Pour les lignes triphasées transposées symétriques, il suffit d'utiliser les attributs ACLineSegment hérités, qui décrivent les impédances et les admittances en séquence pour la longueur totale du segment.

Problème connu: Les attributs exprimant les impédances et les admittances dans PerLengthSequenceImpedance et dans PhaseImpedanceData utilisent Resistance, etc., qui décrivent la longueur totale de segment précalculée, alors qu'il convient d'avoir une unité longitudinale, linéique. En prenant 'r' comme exemple, son unité est 'unit'=Ohm, mais la valeur est effectivement en Ohm/m, si bien qu'il est nécessaire de multiplier la valeur par Conductor.length. Cela va à l'encontre de l'idée globale de types de données d'unité et est incorrect du point de vue sémantique, mais le CIM de base ne contient pas pour l'instant les types de données requis. Jusqu'à la révision de la modélisation des unités dans le CIM, il est nécessaire que les applications déduisent et gèrent localement "/m" d'accolement pour les unités et s'assurent de multiplier les valeurs par Conductor.length.

Le Tableau 33 montre tous les attributs de DistributionLineSegment.

Tableau 33 – Attributs de WiresExt::DistributionLineSegment

nom	type	description
b0ch	Susceptance	hérité de: ACLineSegment
bch	Susceptance	hérité de: ACLineSegment
g0ch	Conductance	hérité de: ACLineSegment
gch	Conductance	hérité de: ACLineSegment
r	Resistance	hérité de: ACLineSegment
r0	Resistance	hérité de: ACLineSegment
X	Reactance	hérité de: ACLineSegment
x0	Reactance	hérité de: ACLineSegment
length	Length	hérité de: Conductor
phases	PhaseCode	hérité de: ConductingEquipment
normallyInService	Boolean	hérité de: Equipment
aliasName	String	hérité de: IdentifiedObject
description	String	hérité de: IdentifiedObject
localName	String	hérité de: IdentifiedObject
mRID	String	hérité de: IdentifiedObject
name	String	hérité de: IdentifiedObject
pathName	String	hérité de: IdentifiedObject

Le Tableau 34 montre toutes les extrémités d'association de DistributionLineSegment avec les autres classes.

Tableau 34 – Extrémités d'association de WiresExt:: DistributionLineSegment avec les autres classes

[mult à partir de]	[mult vers] nom	type	description
[0..*]	[0..1] PhaseImpedance	PerLengthPhaseImpedance	Impédance de phase de ce segment de conducteur; utilisée pour le modèle asymétrique.
[0..*]	[0..1] SequenceImpedance	PerLengthSequenceImpedance	Impédance en séquence de ce segment de conducteur; utilisée pour le modèle symétrique.
[0..*]	[0..1] ConductorInfo	ConductorInfo	Données relatives au conducteur pour ce segment de conducteur.
[0..*]	[0..1] BaseVoltage	BaseVoltage	hérité de: ConductingEquipment
[1..1]	[0..*] Terminals	Terminal	hérité de: ConductingEquipment
[0..*]	[0..*] ProtectionEquipments	ProtectionEquipment	hérité de: ConductingEquipment
[1..1]	[0..*] ClearanceTags	ClearanceTag	hérité de: ConductingEquipment
[1..1]	[0..1] SvStatus	SvStatus	hérité de: ConductingEquipment
[0..*]	[0..1] EquipmentContainer	EquipmentContainer	hérité de: Equipment
[1..1]	[0..*] OperationalLimitSet	OperationalLimitSet	hérité de: Equipment
[1..1]	[0..*] ContingencyEquipment	ContingencyEquipment	hérité de: Equipment
[0..*]	[0..1] PSRType	PSRType	hérité de: PowerSystemResource

[mult à partir de]	[mult vers] nom	type	description
[0..1]	[0..*] Measurements	Measurement	hérité de: PowerSystemResource
[1..1]	[0..*] OperatingShare	OperatingShare	hérité de: PowerSystemResource
[0..*]	[0..*] PsrLists	PsrList	hérité de: PowerSystemResource
[1..1]	[0..1] OutageSchedule	OutageSchedule	hérité de: PowerSystemResource
[0..*]	[0..*] ReportingGroup	ReportingGroup	hérité de: PowerSystemResource
[0..*]	[0..1] Location	Location	hérité de: PowerSystemResource

6.2.3.3 DistributionTapChanger

Paramètres supplémentaires de changeur de prise de rapport communs à des régulateurs de ligne de distribution 'tculControlMode' est toujours 'volt'.

Si 'monitoredPhase' n'est pas spécifié, alors si l'enroulement commandé DistributionTransformerWinding est monophasé, le primaire du transformateur PT est censé être relié au travers de cet enroulement, ce qui est le cas normal. Si l'enroulement commandé est triphasé, 'monitoredPhase' est censé être 'AN', sauf spécification contraire.

Chaque fois que 'ctRatio' et 'ptRatio' sont spécifiés, il est habituel de spécifier R et X en "volts" rapportés au circuit secondaire du PT, autrement R et X sont en ohms du primaire d'alimentation.

Si 'ptRatio' n'est pas spécifié, 'targetVoltage', 'limitVoltage' et 'bandVoltage' sont sur la base primaire d'alimentation, phase-neutre ou phase-phase en fonction de 'monitoredPhase'. Autrement, ces attributs sont tous sur la base secondaire du PT.

Le Tableau 35 montre tous les attributs de DistributionTapChanger.

Tableau 35 – Attributs de WiresExt::DistributionTapChanger

nom	type	description
ctRatio	Float	Rapport du transducteur de courant intégré.
ptRatio	Float	Rapport du transducteur de tension intégré.
reverseLineDropR	Resistance	Réglage de la résistance de compensation de perte de ligne pour le flux de puissance inverse.
reverseLineDropX	Reactance	Réglage de la réactance de compensation de perte de ligne pour le flux de puissance inverse.
lineDropCompensation	Boolean	Si true (vrai), la compensation de perte de linge doit être appliquée.
lineDropR	Resistance	Réglage de la résistance de compensation de perte de ligne pour le flux de puissance normal (direct).
lineDropX	Reactance	Réglage de la réactance de compensation de perte de ligne pour le flux de puissance normal (direct).
targetVoltage	Voltage	Tension cible sur la base secondaire du PT.
bandVoltage	Voltage	Gamme de tensions (max - min) sur la base secondaire du PT, centrée sur 'targetVoltage'.

nom	type	description
limitVoltage	Voltage	Tension régulée maximale autorisée sur la base secondaire du PT, quelle que soit la compensation de perte de ligne. Parfois appelée «first-house protection» (protection de premier stade).
monitoredPhase	PhaseCode	Tension de phase commandant ce régulateur, mesurée à l'emplacement du régulateur.
ctRating	CurrentFlow	Caractéristique assignée du primaire du transformateur de courant intégré
tculControlMode	TransformerControlMode	hérité de: RatioTapChanger
highStep	Integer	hérité de: TapChanger
initialDelay	Seconds	hérité de: TapChanger
lowStep	Integer	hérité de: TapChanger
neutralStep	Integer	hérité de: TapChanger
neutralU	Voltage	hérité de: TapChanger
normalStep	Integer	hérité de: TapChanger
stepVoltageIncrement	PerCent	hérité de: TapChanger
subsequentDelay	Seconds	hérité de: TapChanger
ltcFlag	Boolean	hérité de: TapChanger
regulationStatus	Boolean	hérité de: TapChanger
aliasName	String	hérité de: IdentifiedObject
description	String	hérité de: IdentifiedObject
localName	String	hérité de: IdentifiedObject
mRID	String	hérité de: IdentifiedObject
name	String	hérité de: IdentifiedObject
pathName	String	hérité de: IdentifiedObject

Le Tableau 36 montre toutes les extrémités d'association de DistributionTapChanger avec les autres classes.

Tableau 36 – Extrémités d'association de WiresExt:: DistributionTapChanger avec les autres classes

[mult à partir de]	[mult vers] nom	type	description
[0..1]	[1..1] TransformerWinding	TransformerWinding	hérité de: RatioTapChanger
[1..1]	[0..1] RatioVariationCurve	RatioVariationCurve	hérité de: RatioTapChanger
[0..1]	[1..1] Winding	DistributionTransformerWinding	hérité de: RatioTapChanger
[0..*]	[0..1] RegulatingControl	RegulatingControl	hérité de: TapChanger
[1..1]	[0..1] SvTapStep	SvTapStep	hérité de: TapChanger
[1..1]	[0..1] ImpedanceVariationCurve	ImpedanceVariationCurve	hérité de: TapChanger
[1..1]	[0..*] TapSchedules	TapSchedule	hérité de: TapChanger
[0..*]	[0..1] PSRType	PSRType	hérité de: PowerSystemResource
[0..1]	[0..*] Measurements	Measurement	hérité de: PowerSystemResource
[1..1]	[0..*] OperatingShare	OperatingShare	hérité de: PowerSystemResource
[0..*]	[0..*] PsrLists	PsrList	hérité de: PowerSystemResource

[mult à partir de]	[mult vers] nom	type	description
[1..1]	[0..1] OutageSchedule	OutageSchedule	hérité de: PowerSystemResource
[0..*]	[0..*] ReportingGroup	ReportingGroup	hérité de: PowerSystemResource
[0..*]	[0..1] Location	Location	hérité de: PowerSystemResource

6.2.3.4 DistributionTransformer

Ensemble de deux ou plusieurs enroulements couplés qui transforme la puissance électrique entre des niveaux de tension. Prend en charge les connexions d'enroulements tant équilibrés que déséquilibrés.

Cette classe diffère de Wires::PowerTransformer comme suit:

– elle fait partie de TransformerBank

– elle tire des paramètres exclusivement de TransformerInfo et de ses classes associées.

Le Tableau 37 montre tous les attributs de DistributionTransformer.

Tableau 37 – Attributs de WiresExt::DistributionTransformer

nom	type	description
normallyInService	Boolean	hérité de: Equipment
aliasName	String	hérité de: IdentifiedObject
description	String	hérité de: IdentifiedObject
localName	String	hérité de: IdentifiedObject
mRID	String	hérité de: IdentifiedObject
name	String	hérité de: IdentifiedObject
pathName	String	hérité de: IdentifiedObject

Le Tableau 38 montre toutes les extrémités d'association de DistributionTransformer avec les autres classes.

Tableau 38 – Extrémités d'association de WiresExt:: DistributionTransformer avec les autres classes

[mult à partir de]	[mult vers] nom	type	description
[1..*]	[1..1] TransformerBank	TransformerBank	Batterie à laquelle ce transformateur appartient.
[1..1]	[1..*] Windings	DistributionTransformerWinding	Tous les enroulements de ce transformateur.
[0..*]	[0..1] TransformerInfo	TransformerInfo	Données relatives au transformateur.
[0..*]	[0..1] EquipmentContainer	EquipmentContainer	hérité de: Equipment
[1..1]	[0..*] OperationalLimitSet	OperationalLimitSet	hérité de: Equipment
[1..1]	[0..*] ContingencyEquipment	ContingencyEquipment	hérité de: Equipment
[0..*]	[0..1] PSRType	PSRType	hérité de: PowerSystemResource

[mult à partir de]	[mult vers] nom	type	description
[0..1]	[0..*] Measurements	Measurement	hérité de: PowerSystemResource
[1..1]	[0..*] OperatingShare	OperatingShare	hérité de: PowerSystemResource
[0..*]	[0..*] PsrLists	PsrList	hérité de: PowerSystemResource
[1..1]	[0..1] OutageSchedule	OutageSchedule	hérité de: PowerSystemResource
[0..*]	[0..*] ReportingGroup	ReportingGroup	hérité de: PowerSystemResource
[0..*]	[0..1] Location	Location	hérité de: PowerSystemResource

6.2.3.5 DistributionTransformerWinding

Point de connexion conducteur d'un instance d'enroulement de transformateur de distribution/ déséquilibré.

Cette classe diffère de Wires::TransformerWinding comme suit:

– les huit attributs de modèle Pi sont placés dans une classe distincte, à laquelle référence peut être facultativement faite à partir de plusieurs instances d'enroulement.

– les trois attributs de mise à la terre peuvent différer par instance d'enroulement, même pour des enroulements qui utilisent le même TransformerInfo, et de ce fait ils sont maintenus sur DistributionTransformerWinding.

– l'attribut 'windingType' est remplacé par l'attribut 'sequenceNumber' sur la classe WindingInfo.

– tous les autres attributs proviennent de WindingInfo (et de ses relations). TransformerInfo est associé au DistributionTransformer comme données pouvant être référencées et il peut donc être défini une fois et référencé à partir d'instances, au lieu d'être spécifié avec chaque instance.

Le Tableau 39 montre tous les attributs de DistributionTransformerWinding.

Tableau 39 – Attributs de WiresExt::DistributionTransformerWinding

nom	type	description
grounded	Boolean	(pour les connexions Yn et Zn) True (vrai) si le neutre est mis à la terre solidement.
rground	Resistance	(pour les connexions Yn et Zn) Partie résistive de l'impédance neutre lorsque 'grounded' (mis à la terre) est true (vrai).
xground	Reactance	(pour les connexions Yn et Zn) Partie réactive de l'impédance neutre lorsque 'grounded' (mis à la terre) est true (vrai).
phases	PhaseCode	hérité de: ConductingEquipment
normallyInService	Boolean	hérité de: Equipment
aliasName	String	hérité de: IdentifiedObject
description	String	hérité de: IdentifiedObject
localName	String	hérité de: IdentifiedObject
mRID	String	hérité de: IdentifiedObject
name	String	hérité de: IdentifiedObject
pathName	String	hérité de: IdentifiedObject

Le Tableau 40 montre toutes les extrémités d'association de DistributionTransformerWinding avec les autres classes.

Tableau 40 – Extrémités d'association de WiresExt:: DistributionTransformerWinding avec les autres classes

[mult à partir de]	[mult vers] nom	type	description
[0..*]	[0..1] PiImpedance	WindingPiImpedance	(précis pour les transformateurs à deux ou trois enroulements seulement) Impédances de modèle Pi de cet enroulement.
[1..*]	[1..1] Transformer	DistributionTransformer	Transformateur auquel cet enroulement appartient.
[1..1]	[0..1] RatioTapChanger	RatioTapChanger	Changeur de prise pour rapport associé à cet enroulement.
[1..1]	[0..1] PhaseTapChanger	PhaseTapChanger	Changeur de prise pour phase associé à cet enroulement.
[0..*]	[0..1] WindingInfo	WindingInfo	Données pour cet enroulement.
[0..*]	[0..1] BaseVoltage	BaseVoltage	hérité de: ConductingEquipment
[1..1]	[0..*] Terminals	Terminal	hérité de: ConductingEquipment
[0..*]	[0..*] ProtectionEquipments	ProtectionEquipment	hérité de: ConductingEquipment
[1..1]	[0..*] ClearanceTags	ClearanceTag	hérité de: ConductingEquipment
[1..1]	[0..1] SvStatus	SvStatus	hérité de: ConductingEquipment
[0..*]	[0..1] EquipmentContainer	EquipmentContainer	hérité de: Equipment
[1..1]	[0..*] OperationalLimitSet	OperationalLimitSet	hérité de: Equipment
[1..1]	[0..*] ContingencyEquipment	ContingencyEquipment	hérité de: Equipment
[0..*]	[0..1] PSRType	PSRType	hérité de: PowerSystemResource
[0..1]	[0..*] Measurements	Measurement	hérité de: PowerSystemResource
[1..1]	[0..*] OperatingShare	OperatingShare	hérité de: PowerSystemResource
[0..*]	[0..*] PsrLists	PsrList	hérité de: PowerSystemResource
[1..1]	[0..1] OutageSchedule	OutageSchedule	hérité de: PowerSystemResource
[0..*]	[0..*] ReportingGroup	ReportingGroup	hérité de: PowerSystemResource
[0..*]	[0..1] Location	Location	hérité de: PowerSystemResource

6.2.3.6 PerLengthPhaseImpedance

Paramètres linéiques d'impédance et d'admittance pour les lignes à n fils non équilibrées, sous forme de matrice.

Le Tableau 41 montre tous les attributs de PerLengthPhaseImpedance.

Tableau 41 – Attributs de WiresExt::PerLengthPhaseImpedance

nom	type	description
conductorCount	Integer	Nombre de fils de phase, neutres et autres fils retenus. Contraint le nombre d'éléments de matrice et les codes de phase pouvant être utilisés avec cette matrice.
aliasName	String	hérité de: IdentifiedObject
description	String	hérité de: IdentifiedObject
localName	String	hérité de: IdentifiedObject
mRID	String	hérité de: IdentifiedObject
name	String	hérité de: IdentifiedObject
pathName	String	hérité de: IdentifiedObject

Le Tableau 42 montre toutes les extrémités d'association de PerLengthPhaseImpedance avec les autres classes.

Tableau 42 – Extrémités d'association de WiresExt:: PerLengthPhaseImpedance avec les autres classes

[mult à partir de]	[mult vers] nom	type	description
[1..1]	[1..*] PhaseImpedanceData	PhaseImpedanceData	Toutes les données qui appartiennent à cette impédance de phase de conducteur.
[0..1]	[0..*] ConductorSegments	DistributionLineSegment	Tous les segments de conducteur décrits par cette impédance de phase.

6.2.3.7 PerLengthSequenceImpedance

Paramètres linéiques d'impédance et d'admittance en séquence, pour les lignes transposées monophasées, diphasées ou triphasées. Pour les lignes monophasées, définir x=x0=xself. Pour les lignes diphasées, définir x=xs-xm et x0=xs+xm.

Le Tableau 43 montre tous les attributs de PerLengthSequenceImpedance.

Tableau 43 – Attributs de WiresExt::PerLengthSequenceImpedance

nom	type	description
b0ch	Susceptance	Suspectance shunt homopolaire (chargement), par unité de longueur
bch	Susceptance	Suspectance shunt directe (chargement), par unité de longueur
g0ch	Conductance	Conductance shunt homopolaire (chargement), par unité de longueur
gch	Conductance	Conductance shunt directe (chargement), par unité de longueur
r	Resistance	Résistance série directe, par unité de longueur.
r0	Resistance	Résistance série homopolaire, par unité de longueur.
X	Reactance	Réactance série directe, par unité de longueur.
x0	Reactance	Réactance série homopolaire, par unité de longueur.
aliasName	String	hérité de: IdentifiedObject
description	String	hérité de: IdentifiedObject
localName	String	hérité de: IdentifiedObject
mRID	String	hérité de: IdentifiedObject
name	String	hérité de: IdentifiedObject
pathName	String	hérité de: IdentifiedObject

Le Tableau 44 montre toutes les extrémités d'association de PerLengthSequenceImpedance avec les autres classes.

Tableau 44 – Extrémités d'association de WiresExt:: PerLengthSequenceImpedance avec les autres classes

[mult à partir de]	[mult vers] nom	type	description
[0..1]	[0..*] ConductorSegments	DistributionLineSegment	Tous les segments de conducteur décrits par cette impédance en séquence.

6.2.3.8 PhaseImpedanceData

Triplet de valeurs d'éléments des matrices des résistances, des réactances et des suspectances.

Le Tableau 45 montre tous les attributs de PhaseImpedanceData.

Tableau 45 – Attributs de WiresExt::PhaseImpedanceData

nom	type	description
sequenceNumber	Integer	Indice d'élément en colonne, en supposant une matrice symétrique. Plages de 1 à N + N*(N-1)/2.
b	Susceptance	Valeur d'élément de la matrice des suspectances, par unité de longueur.
r	Resistance	Valeur d'élément de la matrice des résistances, par unité de longueur.
X	Reactance	Valeur d'élément de la matrice des réactances, par unité de longueur.

Le Tableau 46 montre toutes les extrémités d'association de PhaseImpedanceData avec les autres classes.

Tableau 46 – Extrémités d'association de WiresExt::PhaseImpedanceData avec les autres classes

[mult à partir de]	[mult vers] nom	type	description
[1..*]	[1..1] PhaseImpedance	PerLengthPhaseImpedance	Impédance de phase de conducteur à laquelle ces données appartiennent.

6.2.3.9 TransformerBank

Ensemble de transformateurs qui sont raccordés ensemble. Pour les transformateurs triphasés, il y a un transformateur par batterie. Pour les batteries de transformateurs monophasés, il y a au moins deux transformateurs par batterie et il n'est pas nécessaire qu'ils soient identiques.

Le Tableau 47 montre tous les attributs de TransformerBank.

Tableau 47 – Attributs de WiresExt::TransformerBank

nom	type	description
vectorGroup	String	Groupe vectoriel de la batterie pour le relaiement protecteur, par exemple Dyn1. Pour les transformateurs déséquilibrés, cela peut ne pas être déterminé simplement à partir des connexions d'enroulements constitutives.
normallyInService	Boolean	hérité de: Equipment
aliasName	String	hérité de: IdentifiedObject
description	String	hérité de: IdentifiedObject
localName	String	hérité de: IdentifiedObject
mRID	String	hérité de: IdentifiedObject
name	String	hérité de: IdentifiedObject
pathName	String	hérité de: IdentifiedObject

Le Tableau 48 montre toutes les extrémités d'association de TransformerBank avec les autres classes.

Tableau 48 – Extrémités d'association de WiresExt:: TransformerBank avec les autres classes

[mult à partir de]	[mult vers] nom	type	description
[1..1]	[1..*] Transformateurs	DistributionTransformer	Tous les transformateurs qui appartiennent à cette batterie.
[0..*]	[0..1] EquipmentContainer	EquipmentContainer	hérité de: Equipment
[1..1]	[0..*] OperationalLimitSet	OperationalLimitSet	hérité de: Equipment
[1..1]	[0..*] ContingencyEquipment	ContingencyEquipment	hérité de: Equipment
[0..*]	[0..1] PSRType	PSRType	hérité de: PowerSystemResource
[0..1]	[0..*] Measurements	Measurement	hérité de: PowerSystemResource
[1..1]	[0..*] OperatingShare	OperatingShare	hérité de: PowerSystemResource
[0..*]	[0..*] PsrLists	PsrList	hérité de: PowerSystemResource
[1..1]	[0..1] OutageSchedule	OutageSchedule	hérité de: PowerSystemResource
[0..*]	[0..*] ReportingGroup	ReportingGroup	hérité de: PowerSystemResource
[0..*]	[0..1] Location	Location	hérité de: PowerSystemResource

6.2.3.10 WindingPiImpedance

Impédance de modèle Pi de transformateur qui reflète précisément l'impédance pour les transformateurs à deux ou trois enroulements. Pour les transformateurs à quatre enroulements ou plus, il faut utiliser TransformerInfo.

Le Tableau 49 montre tous les attributs de WindingPiImpedance.

Tableau 49 – Attributs de WiresExt::WindingPiImpedance

nom	type	description
b	Susceptance	Susceptance de branche magnétique (B mag). La valeur peut être positive ou négative.
b0	Susceptance	Suspectance de branche magnétique homopolaire.
g	Conductance	Conductance de branche magnétique (G mag).
g0	Conductance	Conductance de branche magnétique homopolaire.
r	Resistance	Résistance CC de l'enroulement.
r0	Resistance	Résistance série homopolaire de l'enroulement.
X	Reactance	Réactance série directe de l'enroulement. Pour un transformateurs à deux enroulements, il convient d'entrer la réactance complète sur l'enroulement (haute tension) primaire.
x0	Reactance	Réactance série homopolaire de l'enroulement.
aliasName	String	hérité de: IdentifiedObject
description	String	hérité de: IdentifiedObject
localName	String	hérité de: IdentifiedObject
mRID	String	hérité de: IdentifiedObject
name	String	hérité de: IdentifiedObject
pathName	String	hérité de: IdentifiedObject

Le Tableau 50 montre toutes les extrémités d'association de WindingPiImpedance avec les autres classes.

Tableau 50 – Extrémités d'association de WiresExt:: WindingPiImpedance avec les autres classes

[mult à partir de]	[mult vers] nom	type	description
[0..1]	[0..*] Windings	DistributionTransformerWinding	Tous les enroulements ayant cette impédance Pi.

6.2.4 Paquetage Assets

6.2.4.1 Généralités

Ce paquetage contient les classes d'informations centrales qui soutiennent des applications de gestion de biens avec des classes spécialisées pour les modèles de niveau biens pour les objets (par opposition aux modèles de ressources de système de puissance définis dans le paquetage IEC61970::Wires).

La Figure 32 montre le diagramme logique AssetsInheritance.

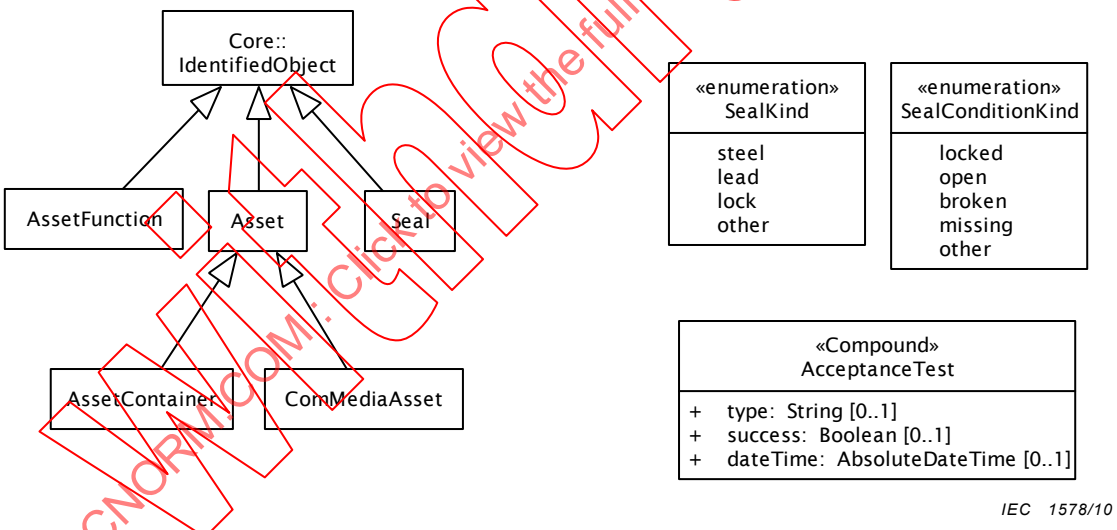


Figure 32 – Diagramme logique Assets::AssetsInheritance.

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 logique AssetsOverview.

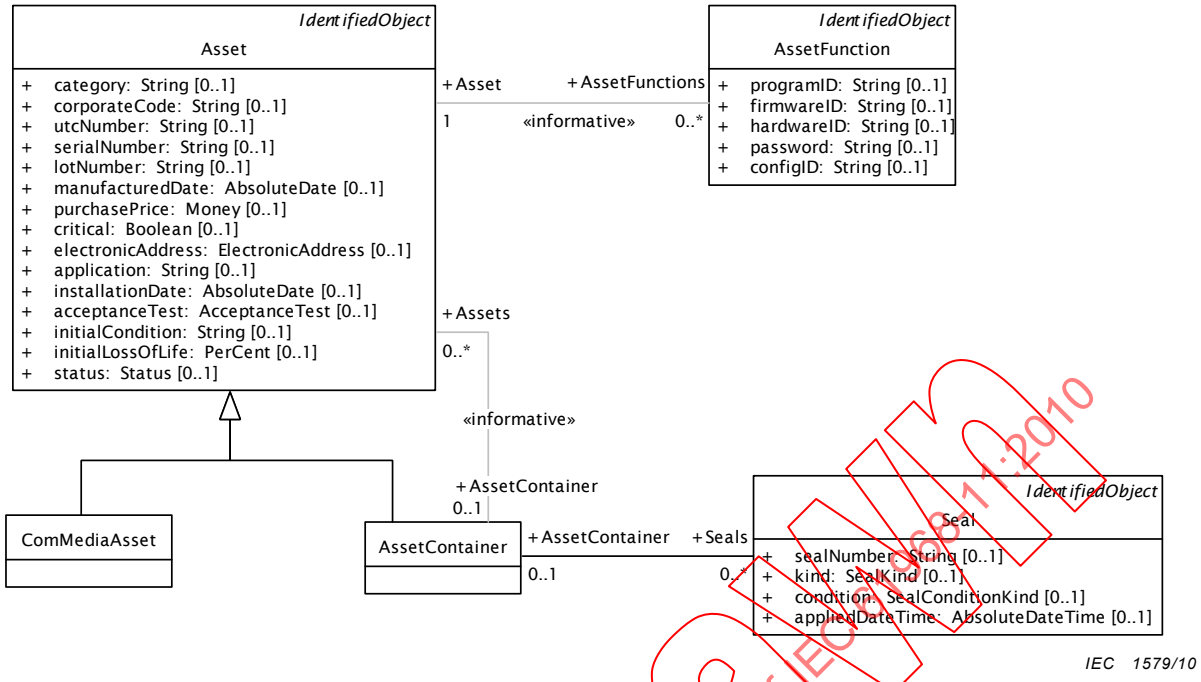


Figure 33 – Diagramme logique Assets::AssetsOverview

Ce diagramme montre les classes normatives issues de ce paquetage.

La Figure 34 montre le diagramme logique DCIMAssetsAndPSRs.

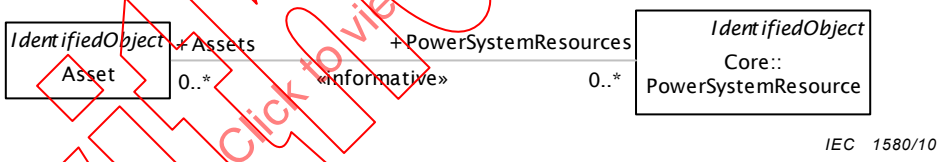


Figure 34 – Diagramme logique Assets::DCIMAssetsAndPSRs

Ce diagramme montre les relations entre la vue ressources de système de puissance et la vue biens de l'équipement dans le DCIM.

6.2.4.2 Compound AcceptanceTest

Essai de réception pour les biens.

Le Tableau 51 montre tous les attributs de AcceptanceTest.

Tableau 51 – 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 (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 (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	AbsoluteDateTime	Date et heure auxquelles l'essai a été soumis à essai la dernière fois en utilisant le 'type' d'essai et fournissant le statut courant dans l'attribut 'success'.

6.2.4.3 Enumération SealConditionKind

Sorte de l'état du joint d'étanchéité.

Le Tableau 52 montre tous les libellés de SealConditionKind.

Tableau 52 – Libellés de Assets::SealConditionKind

libellé	description
locked	
open	
broken	
missing	
other (autre)	

6.2.4.4 Enumération SealKind

Sorte du joint d'étanchéité.

Le Tableau 53 montre tous les libellés de SealKind.

Tableau 53 – Libellés de Assets::SealKind

libellé	description
steel	
lead	
lock	
other (autre)	

6.2.4.5 Asset

Ressource tangible du service public, y compris l'équipement du système de puissance, 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 54 montre tous les attributs de Asset.

Tableau 54 – Attributs de Assets::Asset

nom	type	description
category	String	Mécanisme d'extension pour prendre en charge la 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.)
corporateCode	String	Code pour ce type de bien.
utcNumber	String	Numéro de Uniquely Tracked Commodity (UTC, marchandise localisée de manière univoque).
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.
manufacturedDate	AbsoluteDate	Date de fabrication de ce bien.
purchasePrice	Money	Prix d'achat du bien.
critical	Boolean	True (vrai) si le bien est considéré critique pour une certaine raison (par exemple, un poteau avec des fixations critiques).
application	String	Manière dont ce bien particulier est utilisé dans cette installation. Par exemple, l'application d'une douille lorsqu'il est fixé à un enroulement de transformateur spécifique serait l'un des éléments suivants: H1, H2, H3, H0, X1, X2, X3, X0, Y1, Y2, Y3, Y0.
installationDate	AbsoluteDate	(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).
acceptanceTest	AcceptanceTest	Informations relatives à l'essai de réception.
initialCondition	String	Etat 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.
electronicAddress	ElectronicAddress	Adresse électronique.
aliasName	String	hérité de: IdentifiedObject
description	String	hérité de: IdentifiedObject
localName	String	hérité de: IdentifiedObject
mRID	String	hérité de: IdentifiedObject

nom	type	description
name	String	hérité de: IdentifiedObject
pathName	String	hérité de: IdentifiedObject

Le Tableau 55 montre toutes les extrémités d'association de Asset avec les autres classes.

Tableau 55 – Extrémités d'association de Assets::Asset avec les autres classes

[mult à partir de]	[mult vers] nom	type	description
[0..*]	[0..*] ActivityRecords	ActivityRecord	Tous les enregistrements d'activité créés pour ce bien.

6.2.4.6 AssetContainer

Bien qui est le regroupement d'autres biens tels que conducteurs, transformateurs, dispositifs de commutation, terrain, clôtures, bâtiments, équipements, véhicules, etc.

Le Tableau 56 montre tous les attributs de AssetContainer.

Tableau 56 – Attributs de Assets::AssetContainer

nom	type	description
category	String	hérité de: Asset
corporateCode	String	hérité de: Asset
utcNumber	String	hérité de: Asset
serialNumber	String	hérité de: Asset
lotNumber	String	hérité de: Asset
manufacturedDate	AbsoluteDate	hérité de: Asset
purchasePrice	Money	hérité de: Asset
critical	Boolean	hérité de: Asset
application	String	hérité de: Asset
installationDate	AbsoluteDate	hérité de: Asset
acceptanceTest	AcceptanceTest	hérité de: Asset
initialCondition	String	hérité de: Asset
initialLossOfLife	PerCent	hérité de: Asset
status	Status	hérité de: Asset
electronicAddress	ElectronicAddress	hérité de: Asset
aliasName	String	hérité de: IdentifiedObject
description	String	hérité de: IdentifiedObject
localName	String	hérité de: IdentifiedObject
mRID	String	hérité de: IdentifiedObject
name	String	hérité de: IdentifiedObject
pathName	String	hérité de: IdentifiedObject

Le Tableau 57 montre toutes les extrémités d'association de AssetContainer avec les autres classes.

Tableau 57 – Extrémités d'association 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..*]	[0..*] ActivityRecords	ActivityRecord	hérité de: Asset

6.2.4.7 AssetFunction

Fonction accomplie par un bien.

Le Tableau 58 montre tous les attributs de AssetFunction.

Tableau 58 – 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
description	String	hérité de: IdentifiedObject
localName	String	hérité de: IdentifiedObject
mRID	String	hérité de: IdentifiedObject
name	String	hérité de: IdentifiedObject
pathName	String	hérité de: IdentifiedObject

6.2.4.8 ComMediaAsset

Supports de communication tels que câble de fibre optique, ligne électrique, téléphone, etc.

Le Tableau 59 montre tous les attributs de ComMediaAsset.

Tableau 59 – Attributs de Assets::ComMediaAsset

nom	type	description
category	String	hérité de: Asset
corporateCode	String	hérité de: Asset
utcNumber	String	hérité de: Asset
serialNumber	String	hérité de: Asset
lotNumber	String	hérité de: Asset
manufacturedDate	AbsoluteDate	hérité de: Asset
purchasePrice	Money	hérité de: Asset
critical	Boolean	hérité de: Asset
application	String	hérité de: Asset
installationDate	AbsoluteDate	hérité de: Asset
acceptanceTest	AcceptanceTest	hérité de: Asset
initialCondition	String	hérité de: Asset
initialLossOfLife	PerCent	hérité de: Asset
status	Status	hérité de: Asset
electronicAddress	ElectronicAddress	hérité de: Asset
aliasName	String	hérité de: IdentifiedObject
description	String	hérité de: IdentifiedObject
localName	String	hérité de: IdentifiedObject
mRID	String	hérité de: IdentifiedObject
name	String	hérité de: IdentifiedObject
pathName	String	hérité de: IdentifiedObject

Le Tableau 60 montre toutes les extrémités d'association de ComMediaAsset avec les autres classes.

Tableau 60 – Extrémités d'association de Assets:: ComMediaAsset avec les autres classes

[mult à partir de]	[mult vers] nom	type	description
[0..*]	[0..*] ActivityRecords	ActivityRecord	hérité de: Asset

6.2.4.9 Seal

Contrôle physiquement l'accès aux AssetContainers.

Le Tableau 61 montre tous les attributs de Seal.

Tableau 61 – Attributs de Assets::Seal

nom	type	description
sealNumber	String	(mot réservé) Numéro du joint d'étanchéité
kind	SealKind	Sorte du joint d'étanchéité.
condition	SealConditionKind	État du joint d'étanchéité.
appliedDateTime	AbsoluteDateTime	Date et heures auxquelles ce joint d'étanchéité a été appliqué.
aliasName	String	hérité de: IdentifiedObject
description	String	hérité de: IdentifiedObject
localName	String	hérité de: IdentifiedObject
mRID	String	hérité de: IdentifiedObject
name	String	hérité de: IdentifiedObject
pathName	String	hérité de: IdentifiedObject

Le Tableau 62 montre toutes les extrémités d'association de Seal avec les autres classes.

Tableau 62 – 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 joint d'étanchéité est appliqué.

6.2.5 Paquetage AssetModels

6.2.5.1 Généralités

Ce paquetage est une extension du paquetage Assets et contient les classes d'informations centrales qui soutiennent la gestion des biens et les différentes applications de planification de tâches et de réseau avec des classes de documentation spécialisées décrivant les biens d'un modèle de produit particulier élaboré par un fabricant. Il y a typiquement de nombreuses instances d'un bien associées à un seul modèle de biens. Il contient également des classes *Info "allégées", qui détiennent des attributs de modèle pouvant être référencés non seulement par des Assets mais aussi par des ConductingEquipments.

La Figure 35 montre le diagramme logique AssetModelsInheritance.

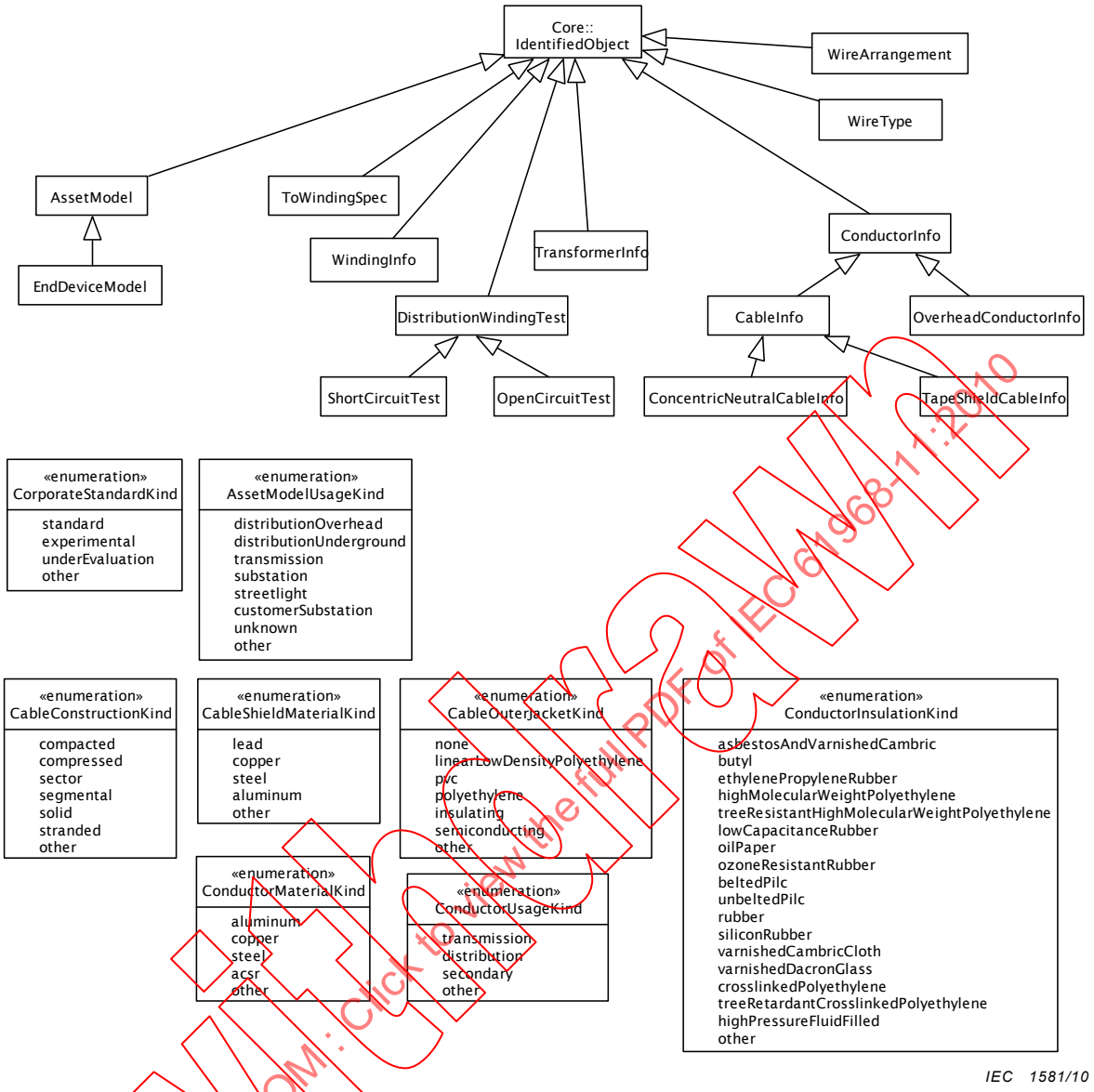


Figure 35 – Diagramme logique AssetModels::AssetModelsInheritance

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 36 montre le diagramme logique AssetModelsOverview.

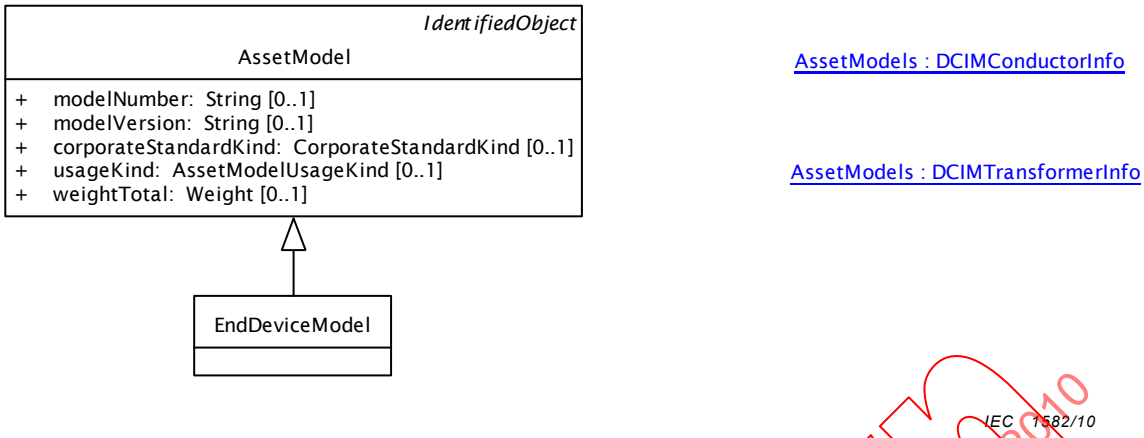


Figure 36 – Diagramme logique AssetModels::AssetModelsOverview

Ce diagramme montre les classes normatives issues de ce paquetage.

La Figure 37 montre le diagramme logique DCIMConductorInfo.

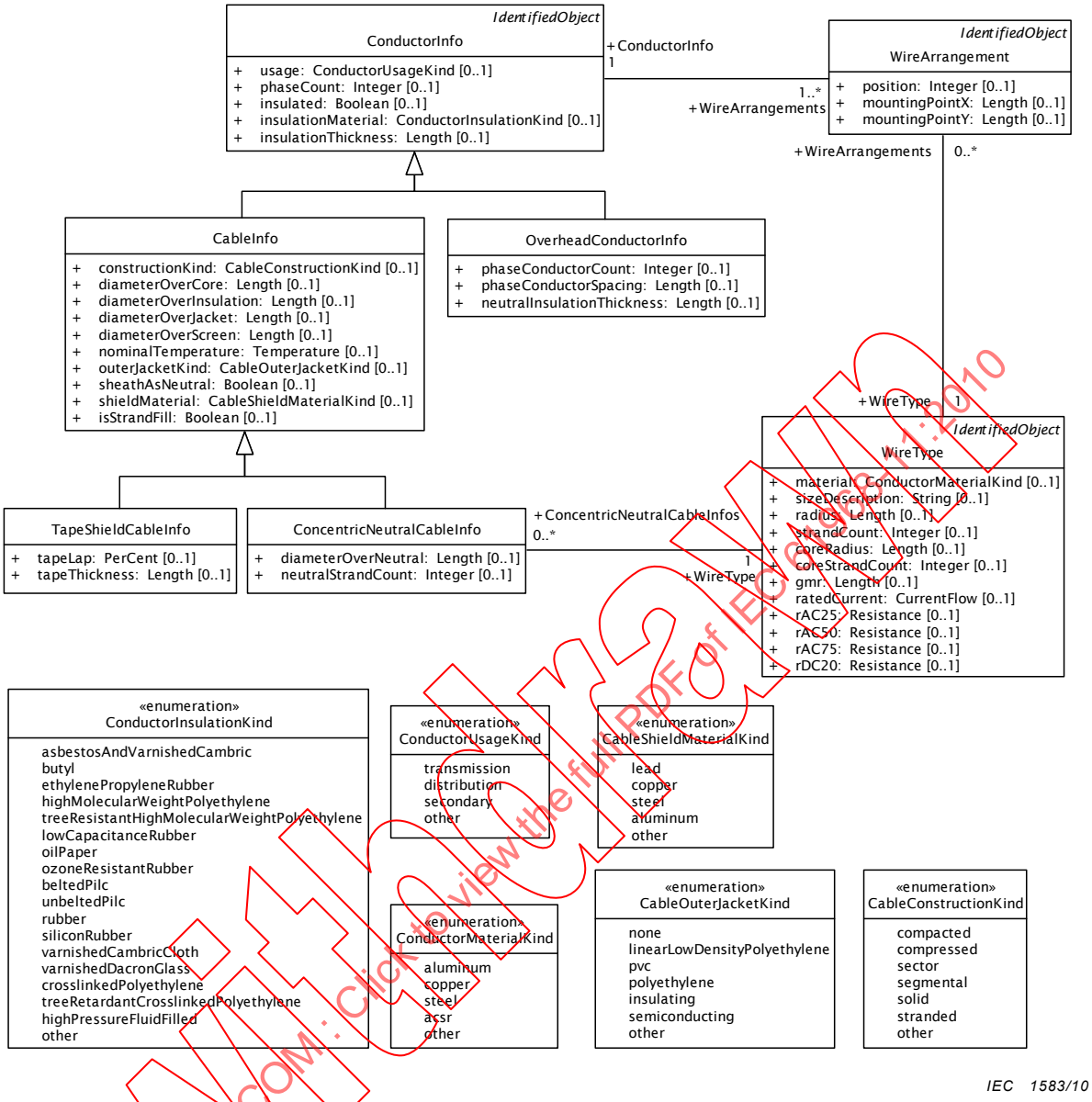


Figure 37 – Diagramme logique AssetModels::DCIMConductorInfo

Ce diagramme montre les classes utilisées pour modéliser les conducteurs dans le DCIM.

La Figure 38 montre le diagramme logique DCIMTransformerInfo.

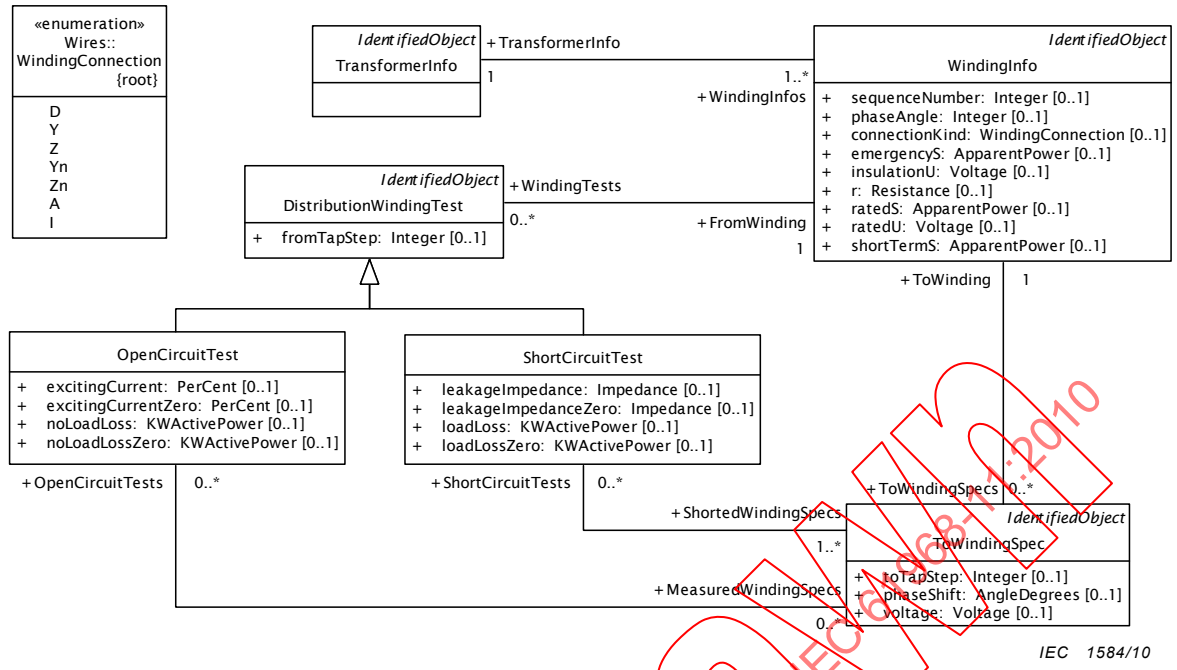


Figure 38 – Diagramme logique AssetModels::DCIMTransformerInfo

Ce diagramme montre une partie des classes utilisées pour modéliser les transformateurs dans le DCIM.

6.2.5.2 Enumération AssetModelUsageKind

Usage pour un modèle de bien.

Le Tableau 63 montre tous les libellés de AssetModelUsageKind.

Tableau 63 – Libellés de AssetModels::AssetModelUsageKind

libellé	description
distributionOverhead	
distributionUnderground	
Transport	
Substation	
streetlight	
customerSubstation	
unknown	
other	

6.2.5.3 Enumération CorporateStandardKind

Sorte de norme d'entreprise.

Le Tableau 64 montre tous les libellés de CorporateStandardKind.

Tableau 64 – Libellés de AssetModels::CorporateStandardKind

libellé	description
Normal	
experimental	
underEvaluation	
other	

6.2.5.4 Enumération CableConstructionKind

Sorte de construction de câble.

Le Tableau 65 montre tous les libellés de CableConstructionKind.

Tableau 65 – Libellés de AssetModels::CableConstructionKind

libellé	description
compacted	
compressed	
sector	
segmental	
solid	
stranded	
other	

6.2.5.5 Enumération CableOuterJacketKind

Sorte de chemise extérieure de câble.

Le Tableau 66 montre tous les libellés de CableOuterJacketKind.

Tableau 66 – Libellés de AssetModels::CableOuterJacketKind

libellé	description
none	
linearLowDensityPolyethylene	
pvc	
polyethylene	
insulating	
semiconducting	
other	

6.2.5.6 Enumération CableShieldMaterialKind

Sorte de matériau de blindage de câble.

Le Tableau 67 montre tous les libellés de CableShieldMaterialKind.

Tableau 67 – Libellés de AssetModels::CableShieldMaterialKind

libellé	description
lead	
copper	
steel	
aluminum	
other	

6.2.5.7 Enumération ConductorInsulationKind

Sorte d'isolant de conducteur.

Le Tableau 68 montre tous les libellés de ConductorInsulationKind.

Tableau 68 – Libellés de AssetModels::ConductorInsulationKind

libellé	description
asbestosAndVarnishedCambric	
butyl	
ethylenePropyleneRubber	
highMolecularWeightPolyethylene	
treeResistantHighMolecularWeightPolyethylene	
lowCapacitanceRubber	
oilPaper	
ozoneResistantRubber	
beltedPilc	
unbeltedPilc	
rubber	
siliconRubber	
varnishedCambricCloth	
varnishedDacronGlass	
crosslinkedPolyethylene	
treeRetardantCrosslinkedPolyethylene	
highPressureFluidFilled	
other	

6.2.5.8 Enumération ConductorMaterialKind

Sorte de matériau de conducteur.

Le Tableau 69 montre tous les libellés de ConductorMaterialKind.

Tableau 69 – Libellés de AssetModels::ConductorMaterialKind

libellé	description
aluminum	
copper	
steel	
acsr	Conducteur en aluminium renforcé d'acier.
other	

6.2.5.9 Enumération ConductorUsageKind

Sorte d'usage de conducteur.

Le Tableau 70 montre tous les libellés de ConductorUsageKind.

Tableau 70 – Libellés de AssetModels::ConductorUsageKind

libellé	description
Transport	
Distribution	
secondary	
other	

6.2.5.10 AssetModel

Documentation pour un modèle de produit particulier élaboré par un fabricant. Il y a typiquement de nombreuses instances d'un bien associées à un seul modèle de biens.

Le Tableau 71 montre tous les attributs de AssetModel.

Tableau 71 – Attributs de AssetModels::AssetModel

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 bien.
usageKind	AssetModelUsageKind	Usage prévu pour ce modèle de bien.
weightTotal	Weight	Poids total manufacturé du bien.
aliasName	String	hérité de: IdentifiedObject
description	String	hérité de: IdentifiedObject
localName	String	hérité de: IdentifiedObject
mRID	String	hérité de: IdentifiedObject
name	String	hérité de: IdentifiedObject
pathName	String	hérité de: IdentifiedObject

6.2.5.11 CableInfo

Données relatives aux câbles.

Le Tableau 72 montre tous les attributs de CableInfo.

Tableau 72 – Attributs de AssetModels::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 la chemise extérieure de ce câble.
sheathAsNeutral	Boolean	True (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 (vrai) si les brins de fils sont extrudés de manière à remplir les vides dans le câble.
usage	ConductorUsageKind	hérité de: ConductorInfo
phaseCount	Integer	hérité de: ConductorInfo
insulationMaterial	ConductorInsulationKind	hérité de: ConductorInfo
insulationThickness	Length	hérité de: ConductorInfo
insulated	Boolean	hérité de: ConductorInfo
aliasName	String	hérité de: IdentifiedObject
description	String	hérité de: IdentifiedObject
localName	String	hérité de: IdentifiedObject
mRID	String	hérité de: IdentifiedObject
name	String	hérité de: IdentifiedObject
pathName	String	hérité de: IdentifiedObject

Le Tableau 73 montre toutes les extrémités d'association de CableInfo avec les autres classes.

Tableau 73 – Extrémités d'association de AssetModels:: CableInfo avec les autres classes

[mult à partir de]	[mult vers] nom	type	description
[1..1]	[1..*] WireArrangements	WireArrangement	hérité de: ConductorInfo
[0..1]	[0..*] ConductorSegments	DistributionLineSegment	hérité de: ConductorInfo

6.2.5.12 ConcentricNeutralCableInfo

Données relatives au câble à neutre concentrique.

Le Tableau 74 montre tous les attributs de ConcentricNeutralCableInfo.

Tableau 74 – Attributs de AssetModels::ConcentricNeutralCableInfo

nom	type	description
diameterOverNeutral	Length	Diamètre sur les brins à neutre concentrique.
neutralStrandCount	Integer	Nombre de brins à neutre concentrique.
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	Température	hérité de: CableInfo
outerJacketKind	CableOuterJacketKind	hérité de: CableInfo
sheathAsNeutral	Boolean	hérité de: CableInfo
shieldMaterial	CableShieldMaterialKind	hérité de: CableInfo
isStrandFill	Boolean	hérité de: CableInfo
usage	ConductorUsageKind	hérité de: ConductorInfo
phaseCount	Integer	hérité de: ConductorInfo
insulationMaterial	ConductorInsulationKind	hérité de: ConductorInfo
insulationThickness	Length	hérité de: ConductorInfo
insulated	Boolean	hérité de: ConductorInfo
aliasName	String	hérité de: IdentifiedObject
description	String	hérité de: IdentifiedObject
localName	String	hérité de: IdentifiedObject
mRID	String	hérité de: IdentifiedObject
name	String	hérité de: IdentifiedObject
pathName	String	hérité de: IdentifiedObject

Le Tableau 75 montre toutes les extrémités d'association de ConcentricNeutralCableInfo avec les autres classes.

Tableau 75 – Extrémités d'association de AssetModels:: ConcentricNeutralCableInfo avec les autres classes

[mult à partir de]	[mult vers] nom	type	description
[0..*]	[1..1] WireType	WireType	Type de fil utilisé pour ce câble à neutre concentrique.
[1..1]	[1..*] WireArrangements	WireArrangement	hérité de: ConductorInfo
[0..1]	[0..*] ConductorSegments	DistributionLineSegment	hérité de: ConductorInfo

6.2.5.13 ConductorInfo

Données relatives au conducteur.

Le Tableau 76 montre tous les attributs de ConductorInfo.

Tableau 76 – Attributs de AssetModels::ConductorInfo

nom	type	description
usage	ConductorUsageKind	Usage de ce conducteur.
phaseCount	Integer	Nombre de phases (neutre compris) à retenir. Il convient de réduire tout fil au-delà de ce nombre en retour par la terre.
insulationMaterial	ConductorInsulationKind	(si conducteur isolé) Matériau utilisé pour l'isolant.
insulationThickness	Length	(si conducteur isolé) Épaisseur de l'isolant.
insulated	Boolean	True (vrai) si le conducteur est isolé.
aliasName	String	hérité de: IdentifiedObject
description	String	hérité de: IdentifiedObject
localName	String	hérité de: IdentifiedObject
mRID	String	hérité de: IdentifiedObject
name	String	hérité de: IdentifiedObject
pathName	String	hérité de: IdentifiedObject

Le Tableau 77 montre toutes les extrémités d'association de ConductorInfo avec les autres classes.

Tableau 77 – Extrémités d'association de AssetModels:: ConductorInfo avec les autres classes

[mult à partir de]	[mult vers] nom	type	description
[1..1]	[1..*] WireArrangements	WireArrangement	Tous les agencements des fils (fils simples) qui composent ce conducteur.
[0..1]	[0..*] ConductorSegments	DistributionLineSegment	Tous les segments de conducteur décrits par ces données relatives au conducteur.

6.2.5.14 DistributionWindingTest

Résultats d'essai pour un ou plusieurs enroulements de transformateur. Ils peuvent comprendre les essais de court-circuit ou de circuit ouvert (excitation). Les résultats d'essai de court-circuit comprennent les pertes par les charges et les impédances de fuite. Les résultats d'essai de circuit ouvert peuvent comprendre les pertes sans charge, le courant d'excitation, les déphasages et la tension induite. Pour les enroulements triphasés, l'excitation peut être directe (la valeur par défaut) ou homopolaire.

Le Tableau 78 montre tous les attributs de DistributionWindingTest.

Tableau 78 – Attributs de AssetModels::DistributionWindingTest

nom	type	description
fromTapStep	Integer	Nombre de régleurs de l'enroulement “from” (départ) de la paire d'essai.
aliasName	String	hérité de: IdentifiedObject
description	String	hérité de: IdentifiedObject
localName	String	hérité de: IdentifiedObject
mRID	String	hérité de: IdentifiedObject
name	String	hérité de: IdentifiedObject
pathName	String	hérité de: IdentifiedObject

Le Tableau 79 montre toutes les extrémités d'association de DistributionWindingTest avec les autres classes.

Tableau 79 – Extrémités d'association de AssetModels:: DistributionWindingTest avec les autres classes

[mult à partir de]	[mult vers] nom	type	description
[0..*]	[1..1] FromWinding	WindingInfo	Enroulement auquel est appliqué(e) la tension ou le courant pendant l'essai.

6.2.5.15 EndDeviceModel

Documentation pour un modèle de produit de dispositif final particulier élaboré par un fabricant.

Le Tableau 80 montre tous les attributs de EndDeviceModel.

Tableau 80 – Attributs de AssetModels::EndDeviceModel

nom	type	description
modelNumber	String	hérité de: AssetModel
modelVersion	String	hérité de: AssetModel
corporateStandardKind	CorporateStandardKind	hérité de: AssetModel
usageKind	AssetModelUsageKind	hérité de: AssetModel
weightTotal	Weight	hérité de: AssetModel
aliasName	String	hérité de: IdentifiedObject
description	String	hérité de: IdentifiedObject
localName	String	hérité de: IdentifiedObject
mRID	String	hérité de: IdentifiedObject
name	String	hérité de: IdentifiedObject
pathName	String	hérité de: IdentifiedObject

Le Tableau 81 montre toutes les extrémités d'association de EndDeviceModel avec les autres classes.

Tableau 81 – Extrémités d'association de AssetModels:: EndDeviceModel avec les autres classes

[mult à partir de]	[mult vers] nom	type	description
[0..1]	[0..*] EndDeviceAssets	EndDeviceAsset	Tous les biens de dispositifs finals qui sont de ce modèle.

6.2.5.16 OpenCircuitTest

Les résultats d'essai de circuit ouvert peuvent comprendre les pertes sans charge, le courant d'excitation, les déphasages et la tension induite. Pour les enroulements triphasés, l'excitation peut être directe (la valeur par défaut) ou homopolaire.

Pour la tension et les déphasages induits, utiliser la classe associée ToWindingSpec.

Le Tableau 82 montre tous les attributs de OpenCircuitTest.

Tableau 82 – Attributs de AssetModels::OpenCircuitTest

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.
noLoadLoss	KWActivePower	Pertes mesurées à partir d'un essai (d'excitation) de circuit ouvert monophasé ou direct.
noLoadLossZero	KWActivePower	Pertes mesurées à partir d'un essai (d'excitation) de circuit ouvert homopolaire.
fromTapStep	Integer	hérité de: DistributionWindingTest
aliasName	String	hérité de: IdentifiedObject
description	String	hérité de: IdentifiedObject
localName	String	hérité de: IdentifiedObject
mRID	String	hérité de: IdentifiedObject
name	String	hérité de: IdentifiedObject
pathName	String	hérité de: IdentifiedObject

Le Tableau 83 montre toutes les extrémités d'association de OpenCircuitTest avec les autres classes.

Tableau 83 – Extrémités d'association de AssetModels:: OpenCircuitTest avec les autres classes

[mult à partir de]	[mult vers] nom	type	description
[0..*]	[0..*] MeasuredWindingSpecs	ToWindingSpec	Tous les autres enroulements mesurés au cours de cet essai.
[0..*]	[1..1] FromWinding	WindingInfo	hérité de: DistributionWindingTest

6.2.5.17 OverheadConductorInfo

Données relatives au conducteur aérien.

Le Tableau 84 montre tous les attributs de OverheadConductorInfo.

Tableau 84 – Attributs de AssetModels::OverheadConductorInfo

nom	type	description
phaseConductorCount	Integer	Nombre de brins conducteurs du faisceau symétrique (1 à 12).
phaseConductorSpacing	Length	Distance entre brins conducteurs dans un faisceau symétrique.
neutralInsulationThickness	Length	(si applicable) Épaisseur de l'isolant du conducteur neutre.
usage	ConductorUsageKind	hérité de: ConductorInfo
phaseCount	Integer	hérité de: ConductorInfo
insulationMaterial	ConductorInsulationKind	hérité de: ConductorInfo
insulationThickness	Length	hérité de: ConductorInfo
insulated	Boolean	hérité de: ConductorInfo
aliasName	String	hérité de: IdentifiedObject
description	String	hérité de: IdentifiedObject
localName	String	hérité de: IdentifiedObject
mRID	String	hérité de: IdentifiedObject
name	String	hérité de: IdentifiedObject
pathName	String	hérité de: IdentifiedObject

Le Tableau 85 montre toutes les extrémités d'association de OverheadConductorInfo avec les autres classes.

Tableau 85 – Extrémités d'association de AssetModels:: OverheadConductorInfo avec les autres classes

[mult à partir de]	[mult vers] nom	type	description
[1..1]	[1..*] WireArrangements	WireArrangement	hérité de: ConductorInfo
[0..1]	[0..*] ConductorSegments	DistributionLineSegment	hérité de: ConductorInfo

6.2.5.18 ShortCircuitTest

Les résultats d'essai de court-circuit 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 ("to") en court-circuit.

Le Tableau 86 montre tous les attributs de ShortCircuitTest.

Tableau 86 – Attributs de AssetModels::ShortCircuitTest

nom	type	description
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 direct.
loadLoss	KWActivePower	Pertes par les charges à partir d'un essai de court-circuit monophasé ou direct.
loadLossZero	KWActivePower	Pertes par les charges à partir d'un essai de court-circuit homopolaire.
fromTapStep	Integer	hérité de: DistributionWindingTest
aliasName	String	hérité de: IdentifiedObject
description	String	hérité de: IdentifiedObject
localName	String	hérité de: IdentifiedObject
mRID	String	hérité de: IdentifiedObject
name	String	hérité de: IdentifiedObject
pathName	String	hérité de: IdentifiedObject

Le Tableau 87 montre toutes les extrémités d'association de ShortCircuitTest avec les autres classes.

Tableau 87 – Extrémités d'association de AssetModels:: ShortCircuitTest avec les autres classes

[mult à partir de]	[mult vers] nom	type	description
[0..*]	[1..*] ShortedWindingSpecs	ToWindingSpec	Tous les autres enroulements en court-circuit au cours de cet essai.
[0..*]	[1..1] FromWinding	WindingInfo	hérité de: DistributionWindingTest

6.2.5.19 TapeShieldCableInfo

Données relatives au câble à blindage en ruban.

Le Tableau 88 montre tous les attributs de TapeShieldCableInfo.

Tableau 88 – Attributs de AssetModels::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

nom	type	description
diameterOverScreen	Length	hérité de: CableInfo
nominalTemperature	Température	hérité de: CableInfo
outerJacketKind	CableOuterJacketKind	hérité de: CableInfo
sheathAsNeutral	Boolean	hérité de: CableInfo
shieldMaterial	CableShieldMaterialKind	hérité de: CableInfo
isStrandFill	Boolean	hérité de: CableInfo
usage	ConductorUsageKind	hérité de: ConductorInfo
phaseCount	Integer	hérité de: ConductorInfo
insulationMaterial	ConductorInsulationKind	hérité de: ConductorInfo
insulationThickness	Length	hérité de: ConductorInfo
insulated	Boolean	hérité de: ConductorInfo
aliasName	String	hérité de: IdentifiedObject
description	String	hérité de: IdentifiedObject
localName	String	hérité de: IdentifiedObject
mRID	String	hérité de: IdentifiedObject
name	String	hérité de: IdentifiedObject
pathName	String	hérité de: IdentifiedObject

Le Tableau 89 montre toutes les extrémités d'association de TapeShieldCableInfo avec les autres classes.

Tableau 89 – Extrémités d'association de AssetModels:: TapeShieldCableInfo avec les autres classes

[mult à partir de]	[mult vers] nom	type	description
[1..1]	[1..*] WireArrangements	WireArrangement	hérité de: ConductorInfo
[0..1]	[0..*] ConductorSegments	DistributionLineSegment	hérité de: ConductorInfo

6.2.5.20 ToWindingSpec

Pour les essais de court-circuit, spécifie l'enroulement et la prise pour tous les enroulements en court-circuit.

Pour les essais de circuit-ouvert, spécifie l'enroulement, la prise, la tension induite, et l'angle induit pour tous les éventuels enroulements non excités qui ont mesurés au cours de l'essai. Cela ne s'appliquera pas si seuls le courant d'excitation et les pertes sans charge ont été mesurés.

Le Tableau 90 montre tous les attributs de ToWindingSpec.

Tableau 90 – Attributs de AssetModels::ToWindingSpec

nom	type	description
toTapStep	Integer	Nombre de régleurs de l'enroulement “to” (destination) de la paire d'essai.
phaseShift	AngleDegrees	(si essai de circuit ouvert) Déphasage mesuré sur l'enroulement “to” en circuit ouvert, l'enroulement "from" étant à la tension assignée de l'enroulement "from" et tous les autres en circuit ouvert.
voltage	Voltage	(si essai de circuit ouvert) Tension mesurée sur l'enroulement “to” en circuit ouvert, l'enroulement "from" étant à la tension assignée de l'enroulement "from" et tous les autres en circuit ouvert.
aliasName	String	hérité de: IdentifiedObject
description	String	hérité de: IdentifiedObject
localName	String	hérité de: IdentifiedObject
mRID	String	hérité de: IdentifiedObject
name	String	hérité de: IdentifiedObject
pathName	String	hérité de: IdentifiedObject

Le Tableau 91 montre toutes les extrémités d'association de ToWindingSpec avec les autres classes.

Tableau 91 – Extrémités d'association de AssetModels:: ToWindingSpec avec les autres classes

[mult à partir de]	[mult vers] nom	type	description
[0..*]	[1..1] ToWinding	WindingInfo	Enroulement en court-circuit dans un essai de court-circuit, ou mesuré pour la tension et l'angle induits dans un essai de circuit ouvert.
[0..*]	[0..*] OpenCircuitTests	OpenCircuitTest	Tous les essais de circuit ouvert dans lesquels cet enroulement a été mesuré.
[1..*]	[0..*] ShortCircuitTests	ShortCircuitTest	Tous les essais de court-circuit dans lesquels cet enroulement a été en court-circuit.

6.2.5.21 TransformerInfo

Jeu de données relatives au transformateur, issues d'une bibliothèque d'équipement.

Le Tableau 92 montre tous les attributs de TransformerInfo.

Tableau 92 – Attributs de AssetModels::TransformerInfo

nom	type	description
aliasName	String	hérité de: IdentifiedObject
description	String	hérité de: IdentifiedObject
localName	String	hérité de: IdentifiedObject
mRID	String	hérité de: IdentifiedObject
name	String	hérité de: IdentifiedObject
pathName	String	hérité de: IdentifiedObject

Le Tableau 93 montre toutes les extrémités d'association de TransformerInfo avec les autres classes.

Tableau 93 – Extrémités d'association de AssetModels:: TransformerInfo avec les autres classes

[mult à partir de]	[mult vers] nom	type	description
[1..1]	[1..*] WindingInfos	WindingInfo	Données pour tous les enroulements décrits par ces données relatives au transformateur.
[0..1]	[0..*] Transformateurs	DistributionTransformer	Tous les transformateurs qui peuvent être décrits par ces données relatives au transformateur.

6.2.5.22 WindingInfo

Données relatives à l'enroulement.

Le Tableau 94 montre tous les attributs de WindingInfo.

Tableau 94 – Attributs de AssetModels::WindingInfo

nom	type	description
phaseAngle	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: 'connectionKind' = Yn et 'phaseAngle' = 11.
connectionKind	WindingConnection	Sorte de la connexion de cet enroulement.
emergencyS	ApparentPower	Puissance apparente pouvant être supporté par l'enroulement dans des conditions d'urgence.
insulationU	Voltage	Puissance nominale de tension au niveau d'isolement de base.
r	Resistance	Résistance CC de cet enroulement.
ratedS	ApparentPower	Puissance apparente assignée normale de cet enroulement.
ratedU	Voltage	Tension assignée de cet enroulement: entre phases pour les enroulements triphasés et soit entre phases, soit entre phase et neutre pour les enroulements monophasés.

nom	type	description
shortTermS	ApparentPower	Puissance apparente pouvant être supporté par cet enroulement sur une courte période.
sequenceNumber	Integer	Numéro de séquence pour cet enroulement, correspondante à l'ordre de l'enroulement dans l'attribut TransformerBank.vectorGroup. Il convient que l'enroulement de plus haute tension soit le '1'.
aliasName	String	hérité de: IdentifiedObject
description	String	hérité de: IdentifiedObject
localName	String	hérité de: IdentifiedObject
mRID	String	hérité de: IdentifiedObject
name	String	hérité de: IdentifiedObject
pathName	String	hérité de: IdentifiedObject

Le Tableau 95 montre toutes les extrémités d'association de WindingInfo avec les autres classes.

Tableau 95 – Extrémités d'association de AssetModels:: WindingInfo avec les autres classes

[mult à partir de]	[mult vers] nom	type	description
[1..1]	[0..*] WindingTests	DistributionWindingTest	Tous les essais d'enroulement pendant lesquels la tension ou le courant a été appliqué(e) à cet enroulement.
[0..1]	[0..*] Windings	DistributionTransformerWinding	Tous les enroulements décrits par ces données relatives aux enroulements.
[1..1]	[0..*] ToWindingSpecs	ToWindingSpec	Régleurs et mesures de tension/d'angle induit(e) pour les essais dans lesquels cet enroulement n'a pas été excité.
[1..*]	[1..1] TransformerInfo	TransformerInfo	Données relatives au transformateur dont la description de cet enroulement fait partie.

6.2.5.23 WireArrangement

Identification, espacement et configuration des fils d'un Conductor (conducteur) en fonction de leur type.

Le Tableau 96 montre tous les attributs de WireArrangement.

Tableau 96 – Attributs de AssetModels::WireArrangement

nom	type	description
position	Integer	Numéro de position sur la structure correspondant à ce fil. Par exemple, utiliser 1..3 pour les phases et 4 pour le neutre sur une structure triphasée. Les affectations de phase individuelles importent; par exemple, ABC produira un jeu de paramètres de ligne déséquilibrés, par phase, différent de celui produit par BAC.
mountingPointX	Length	Distance horizontale signée allant du premier fil au point de référence commun.
mountingPointY	Length	Hauteur au-dessus du sol du premier fil.
aliasName	String	hérité de: IdentifiedObject
description	String	hérité de: IdentifiedObject
localName	String	hérité de: IdentifiedObject
mRID	String	hérité de: IdentifiedObject
name	String	hérité de: IdentifiedObject
pathName	String	hérité de: IdentifiedObject

Le Tableau 97 montre toutes les extrémités d'association de WireArrangement avec les autres classes.

Tableau 97 – Extrémités d'association de AssetModels:: WireArrangement avec les autres classes

[mult à partir de]	[mult vers] nom	type	description
[0..*]	[1..1] WireType	WireType	Type de fils utilisé pour cet agencement de fils.
[1..*]	[1..1] ConductorInfo	ConductorInfo	Données relatives au conducteur auxquelles cet agencement de fils appartient.

6.2.5.24 WireType

Fil conducteur (selon les spécifications de l'IEEE). Type spécifique de fil ou de combinaison de fils, non isolés les uns des autres, capable de transporter le courant électrique.

Le Tableau 98 montre tous les attributs de WireType.

Tableau 98 – Attributs de AssetModels::WireType

nom	type	description
material	ConductorMaterialKind	Matériau du fil.
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 fil.
coreRadius	Length	(en présence de matériau différent pour l'âme) Rayon de l'âme centrale.

nom	type	description
coreStrandCount	Integer	(si utilisé) Nombre de brins dans l'âme d'acier.
gmr	Length	Rayon moyen géométrique (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	Resistance	Résistance CA linéique du conducteur à 25 °C.
rAC50	Resistance	Résistance CA linéique du conducteur à 50 °C.
rAC75	Resistance	Résistance CA linéique du conducteur à 75 °C.
rDC20	Resistance	Résistance CC linéique du conducteur à 20 °C.
aliasName	String	hérité de: IdentifiedObject
description	String	hérité de: IdentifiedObject
localName	String	hérité de: IdentifiedObject
mRID	String	hérité de: IdentifiedObject
name	String	hérité de: IdentifiedObject
pathName	String	hérité de: IdentifiedObject

Le Tableau 99 montre toutes les extrémités d'association de WireType avec les autres classes.

Tableau 99 – Extrémités d'association de AssetModels:: WireType avec les autres classes

[mult à partir de]	[mult vers] nom	type	description
[1..1]	[0..*] ConcentricNeutralCableInfos	ConcentricNeutralCableInfo	Tous les câbles à neutre concentrique utilisant ce type de fil.
[1..1]	[0..*] WireArrangements	WireArrangement	Tous les agencements de fils utilisant ce type de fil.

6.2.6 Paquetage Work

6.2.6.1 Généralités

Ce paquetage contient les classes d'informations centrales qui soutiennent les applications de gestion de travaux et de planification d'extension de réseau.

La Figure 39 montre le diagramme logique WorkInheritance.

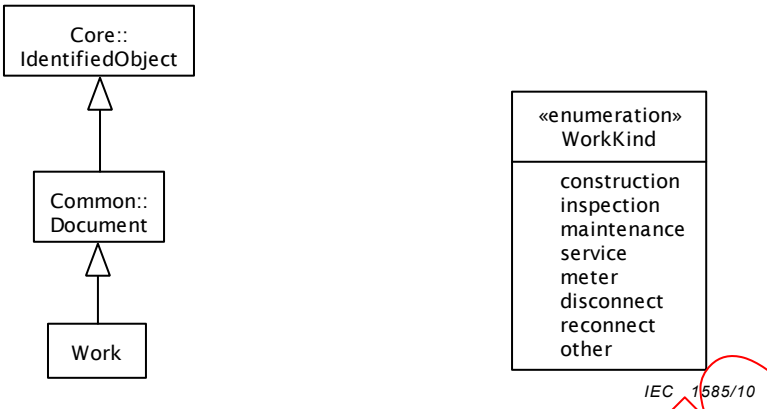


Figure 39 – Diagramme logique Work::WorkInheritance

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 40 montre le diagramme logique WorkOverview.

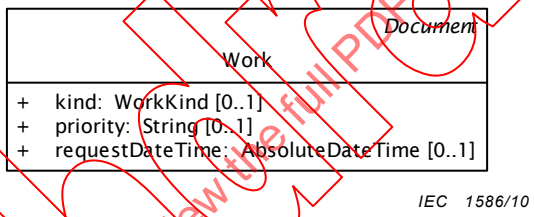


Figure 40 – Diagramme logique Work::WorkOverview

Ce diagramme montre les classes normatives issues de ce paquetage.

6.2.6.2 Enumération WorkKind

Sorte des travaux.

Le Tableau 100 montre tous les libellés de WorkKind.

Tableau 100 – Libellés de Work::WorkKind

libellé	description
construction	
inspection	
maintenance	
service	
meter	
disconnect	
reconnect	
other	

6.2.6.3 Work

Document utilisé pour demander, déclencher, suivre et enregistrer des travaux. Il est synonyme de Work Breakdown Structure (WBS, structure de répartition du travail), qui est traversée par l'association récursive (actuellement informative) de Work.

Remarquer que le nom de la tâche est égal au nom WBS, qui est donné dans l'attribut hérité "name".

Le Tableau 101 montre tous les attributs de Work.

Tableau 101 – Attributs de Work::Work

nom	type	description
kind	WorkKind	Sorte des travaux.
priority	String	Priorité des travaux.
requestDateTime	AbsoluteDateTime	Date et heure à laquelle le travail est demandé.
category	String	hérité de: Document
createdDateTime	AbsoluteDateTime	hérité de: Document
lastModifiedDateTime	AbsoluteDateTime	hérité de: Document
revisionNumber	String	hérité de: Document
subject	String	hérité de: Document
title	String	hérité de: Document
docStatus	Status	hérité de: Document
status	Status	hérité de: Document
electronicAddress	ElectronicAddress	hérité de: Document
aliasName	String	hérité de: IdentifiedObject
description	String	hérité de: IdentifiedObject
localName	String	hérité de: IdentifiedObject
mRID	String	hérité de: IdentifiedObject
name	String	hérité de: IdentifiedObject
pathName	String	hérité de: IdentifiedObject

Le Tableau 102 montre toutes les extrémités d'association de Work avec les autres classes.

Tableau 102 – Extrémités d'association 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..*]	[0..*] ActivityRecords	ActivityRecord	hérité de: Document

6.2.7 Paquetage Customers

6.2.7.1 Généralités

Ce paquetage contient les classes d'informations centrales qui soutiennent les applications de facturation client.

La Figure 41 montre le diagramme logique CustomersInheritance.

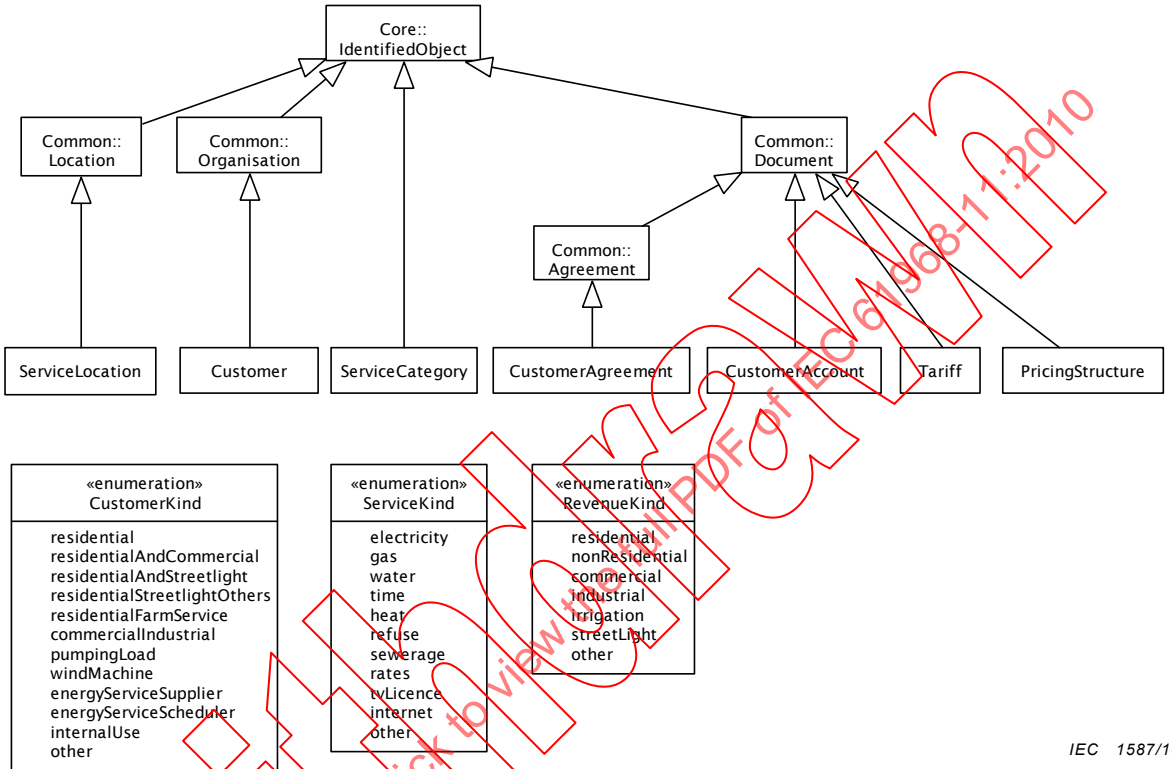


Figure 41 – Diagramme logique Customers::CustomersInheritance

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 42 montre le diagramme logique CustomersOverview.

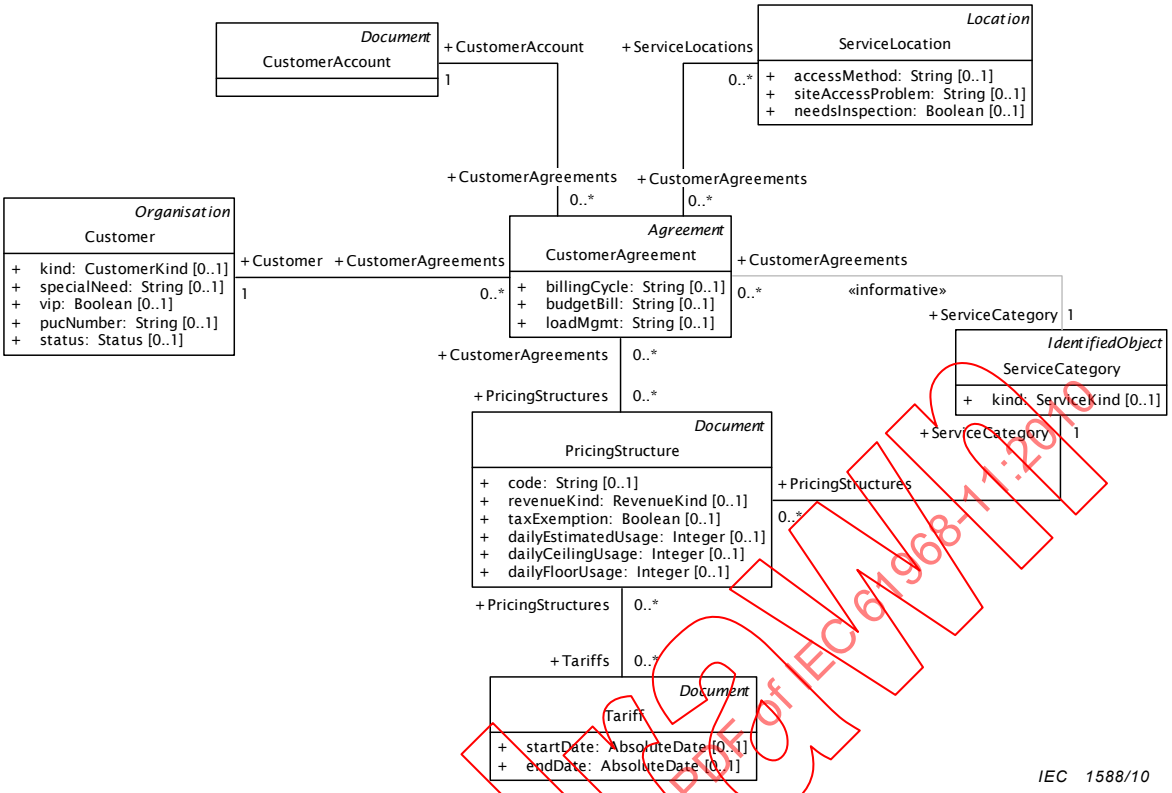


Figure 42 – Diagramme logique Customers::CustomersOverview

Ce diagramme montre les classes normatives issues de ce paquetage.

6.2.7.2 Enumération CustomerKind

Sorte de client.

Le Tableau 103 montre tous les libellés de CustomerKind.

Tableau 103 – Libellés de Customers::CustomerKind

libellé	description
residential	
residentialAndCommercial	
residentialAndStreetlight	
residentialStreetlightOthers	
residentialFarmService	
commercialIndustrial	
pumpingLoad	
windMachine	
energyServiceSupplier	
energyServiceScheduler	
internalUse	
other	

6.2.7.3 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 104 montre tous les libellés de RevenueKind.

Tableau 104 – Libellés de Customers::RevenueKind

libellé	description
residential	
nonResidential	
commercial	
industrial	
irrigation	
streetLight	
other	

6.2.7.4 Enumération ServiceKind

Sorte de service.

Le Tableau 105 montre tous les libellés de ServiceKind.

Tableau 105 – Libellés de Customers::ServiceKind

libellé	description
electricity	
gas	
water	
time	
heat	
refuse	
sewerage	
rates	
tvLicence	
internet	
other	

6.2.7.5 Customer

Organisation recevant des services de la part de ServiceSupplier.

Le Tableau 106 montre tous les attributs de Customer.

Tableau 106 – Attributs de Customers::Customer

nom	type	description
kind	CustomerKind	Sorte de client.
specialNeed	String	True (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 (Public Utility Commission, PUC).
status	Status	Statut de ce client.
streetAddress	StreetAddress	hérité de: Organisation
postalAddress	PostalAddress	hérité de: Organisation
phone1	TelephoneNumber	hérité de: Organisation
phone2	TelephoneNumber	hérité de: Organisation
electronicAddress	ElectronicAddress	hérité de: Organisation
aliasName	String	hérité de: IdentifiedObject
description	String	hérité de: IdentifiedObject
localName	String	hérité de: IdentifiedObject
mRID	String	hérité de: IdentifiedObject
name	String	hérité de: IdentifiedObject
pathName	String	hérité de: IdentifiedObject

Le Tableau 107 montre toutes les extrémités d'association de Customer avec les autres classes.

Tableau 107 – Extrémités d'association de Customers:: Customer avec les autres classes

[mult à partir de]	[mult vers] nom	type	description
[0..1]	[0..*] EndDeviceAssets	EndDeviceAsset	Tous les biens de dispositifs finals de ce client.
[1..1]	[0..*] CustomerAgreements	CustomerAgreement	Tous les accords de ce client.
[0..*]	[0..*] Works	Work	Toutes les tâches accomplies pour ce client.

6.2.7.6 CustomerAccount

Affectation d'un groupe de produits et de services achetés par le client Customer par un accord client CustomerAgreement, utilisée comme mécanisme de facturation et paiement client. Il contient les informations communes issues de divers types d'accords CustomerAgreements pour créer des facturations (factures) pour un client Customer et recevoir un paiement.

Le Tableau 108 montre tous les attributs de CustomerAccount.

Tableau 108 – Attributs de Customers::CustomerAccount

nom	type	description
category	String	hérité de: Document
createdDateTime	AbsoluteDateTime	hérité de: Document
lastModifiedDateTime	AbsoluteDateTime	hérité de: Document
revisionNumber	String	hérité de: Document
subject	String	hérité de: Document
title	String	hérité de: Document
docStatus	Status	hérité de: Document
status	Status	hérité de: Document
electronicAddress	ElectronicAddress	hérité de: Document
aliasName	String	hérité de: IdentifiedObject
description	String	hérité de: IdentifiedObject
localName	String	hérité de: IdentifiedObject
mRID	String	hérité de: IdentifiedObject
name	String	hérité de: IdentifiedObject
pathName	String	hérité de: IdentifiedObject

Le Tableau 109 montre toutes les extrémités d'association de CustomerAccount avec les autres classes.

Tableau 109 – Extrémités d'association 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	Toutes les accords pour ce compte client.
[0..*]	[0..*] ActivityRecords	ActivityRecord	hérité de: Document

6.2.7.7 CustomerAgreement

Accord entre le client Customer et le prestataire de services ServiceSupplier pour payer le service en un emplacement spécifique de service ServiceLocation. 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 110 montre tous les attributs de CustomerAgreement.

Tableau 110 – Attributs de Customers::CustomerAgreement

nom	type	description
billingCycle	String	Jour de 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.
loadMgmt	String	Code de gestion de charge.
signDate	AbsoluteDate	hérité de: Agreement
validityInterval	DateTimeInterval	hérité de: Agreement
category	String	hérité de: Document
createdDateTime	AbsoluteDateTime	hérité de: Document
lastModifiedDateTime	AbsoluteDateTime	hérité de: Document
revisionNumber	String	hérité de: Document
subject	String	hérité de: Document
title	String	hérité de: Document
docStatus	Status	hérité de: Document
status	Status	hérité de: Document
electronicAddress	ElectronicAddress	hérité de: Document
aliasName	String	hérité de: IdentifiedObject
description	String	hérité de: IdentifiedObject
localName	String	hérité de: IdentifiedObject
mRID	String	hérité de: IdentifiedObject
name	String	hérité de: IdentifiedObject
pathName	String	hérité de: IdentifiedObject

Le Tableau 111 montre toutes les extrémités d'association de CustomerAgreement avec les autres classes.

Tableau 111 – Extrémités d'association 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..*] ServiceDeliveryPoints	ServiceDeliveryPoint	Tous les points de livraison de service régis par cet accord client.
[0..1]	[0..*] MeterReadings	MeterReading	(susceptible d'être déconseillé dans le futur) Tous les relevés de lecture de compteurs pour cet accord client.
[0..*]	[1..1] ServiceSupplier	ServiceSupplier	Fournisseur de service pour cet accord client.
[0..*]	[1..1] CustomerAccount	CustomerAccount	Compte client propriétaire de cet accord.

[mult à partir de]	[mult vers] nom	type	description
[0..*]	[1..1] Customer	Customer	Client pour cet accord.
[0..*]	[0..*] ActivityRecords	ActivityRecord	hérité de: Document

6.2.7.8 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 (à savoir, structure de tarification des droits, structure de tarification des dépôts, structure de tarification des services d'électricité, etc.) et à la comptabilité.

Le Tableau 112 montre tous les attributs de PricingStructure.

Tableau 112 – 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 utilisé 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.).
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.
category	String	hérité de: Document
createdDateTime	AbsoluteDateTime	hérité de: Document
lastModifiedDateTime	AbsoluteDateTime	hérité de: Document
revisionNumber	String	hérité de: Document
subject	String	hérité de: Document
title	String	hérité de: Document
docStatus	Status	hérité de: Document
status	Status	hérité de: Document
electronicAddress	ElectronicAddress	hérité de: Document
aliasName	String	hérité de: IdentifiedObject
description	String	hérité de: IdentifiedObject