

Energy management system application program interface (EMS-API) —

Part 301: Common Information Model (CIM) Base

The European Standard EN 61970-301:2004 has the status of a
British Standard

ICS 33.200

National foreword

This British Standard is the official English language version of EN 61970-301:2004. It is identical with IEC 61970-301:2003.

The UK participation in its preparation was entrusted to Technical Committee PEL/57, Power system control and associated communications, which has the responsibility to:

- aid enquirers to understand the text;
- present to the responsible international/European committee any enquiries on the interpretation, or proposals for change, and keep the UK interests informed;
- monitor related international and European developments and promulgate them in the UK.

A list of organizations represented on this committee can be obtained on request to its secretary.

Cross-references

The British Standards which implement international or European publications referred to in this document may be found in the *BSI Catalogue* under the section entitled “International Standards Correspondence Index”, or by using the “Search” facility of the *BSI Electronic Catalogue* or of British Standards Online.

This publication does not purport to include all the necessary provisions of a contract. Users are responsible for its correct application.

Compliance with a British Standard does not of itself confer immunity from legal obligations.

Summary of pages

This document comprises a front cover, an inside front cover, the EN title page, pages 2 to 176, an inside back cover and a back cover.

The BSI copyright notice displayed in this document indicates when the document was last issued.

Amendments issued since publication

Amd. No.	Date	Comments

This British Standard was published under the authority of the Standards Policy and Strategy Committee on 15 April 2004

© BSI 15 April 2004

ISBN 0 580 43644 6

**Energy management system application program interface (EMS-API)
Part 301: Common Information Model (CIM) Base
(IEC 61970-301:2003)**

Système de gestion d'énergie –
Interface de programmation d'application
(EMS-API)
Partie 301: Base de Modèle d'Information
Commun (CIM)
(CEI 61970-301:2003)

Anwendungsprogramm-Schnittstelle für
Netzführungssysteme (EMS-API)
Teil 301: Allgemeines Informationsmodell
(CIM), Basismodell
(IEC 61970-301:2003)

This European Standard was approved by CENELEC on 2004-02-01. CENELEC members are bound to comply with the CEN/CENELEC Internal Regulations which stipulate the conditions for giving this European Standard the status of a national standard without any alteration.

Up-to-date lists and bibliographical references concerning such national standards may be obtained on application to the Central Secretariat or to any CENELEC member.

This European Standard exists in three official versions (English, French, German). A version in any other language made by translation under the responsibility of a CENELEC member into its own language and notified to the Central Secretariat has the same status as the official versions.

CENELEC members are the national electrotechnical committees of Austria, Belgium, Cyprus, Czech Republic, Denmark, Estonia, Finland, France, Germany, Greece, Hungary, Iceland, Ireland, Italy, Latvia, Lithuania, Luxembourg, Malta, Netherlands, Norway, Poland, Portugal, Slovakia, Slovenia, Spain, Sweden, Switzerland and United Kingdom.

CENELEC

European Committee for Electrotechnical Standardization
Comité Européen de Normalisation Electrotechnique
Europäisches Komitee für Elektrotechnische Normung

Central Secretariat: rue de Stassart 35, B - 1050 Brussels

Foreword

The text of document 57/656/FDIS, future edition 1 of IEC 61970-301, prepared by IEC TC 57, Power systems management and associated information exchange, was submitted to the IEC-CENELEC parallel vote and was approved by CENELEC as EN 61970-301 on 2004-02-01.

The following dates were fixed:

- latest date by which the EN has to be implemented
at national level by publication of an identical
national standard or by endorsement (dop) 2004-11-01
- latest date by which the national standards conflicting
with the EN have to be withdrawn (dow) 2007-02-01

The International Electrotechnical Commission (IEC) and CENELEC draw attention to the fact that it is claimed that compliance with this document may involve the use of a patent concerning a computer-based implementation of an object-oriented power system model in a relational database. As such, it does not conflict with the development of any logical power system model including the Common Information Model (CIM), where implementation of the model is not defined.

The IEC and CENELEC take no position concerning the evidence, validity and scope of this patent right.

The holder of this patent right, ICL, has assured the IEC that they are willing to grant a royalty free license to any entity implementing this standard. This license is issued by default, and vendors wishing to take up the license are not required to notify ICL. The statement of the holder of this patent right is registered with the IEC. Information may be obtained from:

ICL
Wenlock WAY
West Gorton
Manchester
M12 5DR
United Kingdom

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights other than those identified above. IEC and CENELEC shall not be held responsible for identifying any or all such patent rights.

Annex ZA has been added by CENELEC.

Endorsement notice

The text of the International Standard IEC 61970-301:2003 was approved by CENELEC as a European Standard without any modification.

CONTENTS

INTRODUCTION.....	5
1 Scope.....	6
2 Normative references	6
3 Terms and definitions	6
4 CIM specification	7
4.1 CIM modeling notation.....	7
4.2 CIM packages	7
4.2.1 Core	9
4.2.2 Topology	9
4.2.3 Wires.....	9
4.2.4 Outage	9
4.2.5 Protection.....	9
4.2.6 Meas	9
4.2.7 LoadModel.....	9
4.2.8 Generation	9
4.2.9 Domain.....	10
4.3 CIM classes and relationships	10
4.3.1 Generalization	11
4.3.2 Simple association	11
4.3.3 Aggregation.....	12
4.4 CIM model concepts and examples	12
4.4.1 Transformer model	12
4.4.2 Connectivity model	13
4.4.3 Inheritance Hierarchy	16
4.4.4 Equipment Containers	18
4.5 Modeling tools.....	18
4.6 Modeling guidelines.....	19
4.6.1 Amendments to the CIM	19
4.6.2 CIM profiles.....	20
4.7 User implementation conventions	20
4.7.1 Naming.....	20
4.7.2 Use of Measurement-related classes	21
4.7.3 Number of Terminals for ConductingEquipment Objects	24
4.8 Examples	24
Annex A (normative) Common information model for control center application program interface	25
Annex B (informative) CIM notation mapping from entity relationship diagram to class diagram in UML	173
Annex ZA (normative) Normative references to international publications with their corresponding European publications	175
Bibliography.....	176

Figure 1 – CIM Part 301 Package Diagram	8
Figure 2 – Example of generalization	11
Figure 3 – Example of Simple Association	12
Figure 4 – Example of Aggregation	12
Figure 5 – Transformer Model	13
Figure 6 – Connectivity Model	14
Figure 7 – Simple Network Example	15
Figure 8 – Simple Network Connectivity Modeled with CIM Topology	16
Figure 9 – Equipment Inheritance Hierarchy	17
Figure 10 – Equipment Containers	18
Figure 11 – Navigating from PSR to MeasurementValue	22
Figure 12 – Measurement placement	23
Figure A.1 – CIM Top Level Packages	25
Figure A.2 – Main	26
Figure A.3 – Main	27
Figure A.4 – Integer Datatypes	38
Figure A.5 – Float Datatypes	39
Figure A.6 – String Datatypes	40
Figure A.7 – Other Datatypes	41
Figure A.8 – Enumeration Datatypes	42
Figure A.9 – Main	62
Figure A.10 – Main	63
Figure A.11 – Hydro	64
Figure A.12 – Thermal	65
Figure A.13 – Main	97
Figure A.14 – Main	110
Figure A.15 – Main	120
Figure A.16 – Measurements	121
Figure A.17 – Quality	122
Figure A.18 – Main	130
Figure A.19 – Main	133
Figure A.20 – Main	136
Figure A.21 – Transformer Model	139
Figure A.22 – EquipmentContainment	140
Figure A.23 – InheritanceHierarchy	141
Figure A.24 – LineModel	142
Figure A.25 – RegulatingEquipment	143
Figure A.26 – VoltageControl	144
Table 1 – MeasurementType Naming Conventions	22
Table 2 – MeasurementValueSource Naming Conventions	23
Table B.1 – CIM Mapping Conventions	173

INTRODUCTION

This standard is part of the IEC 61970 series, which defines an Application Program Interface (API) for an Energy Management System (EMS). This standard is based upon the work of the EPRI Control Center API (CCAPI) research project (RP-3654-1). The principle objectives of the EPRI CCAPI project are to:

- reduce the cost and time needed to add new applications to an EMS;
- protect the investment of existing applications or systems that are working effectively with an EMS.

The principal task of the CCAPI project is to produce requirements and draft text for standards to facilitate the integration of EMS applications developed independently by different vendors, between entire EMS systems developed independently, or between an EMS system and other systems concerned with different aspects of power system operations, such as generation or Distribution Management Systems (DMS). This is accomplished by defining application program interfaces to enable these applications or systems access to public data and exchange information independent of how such information is represented internally. The Common Information Model (CIM) specifies the semantics for this API. The Component Interface Specifications (CIS) specify the content of the messages exchanged.

This part of the series, IEC 61970-301, defines the CIM Base set of packages which provide a logical view of the physical aspects of Energy Management System information. Future IEC 61970-302 defines the financial and energy scheduling logical view. Future IEC 61970-303 defines the SCADA logical view. The CIM is an abstract model that represents all the major objects in an electric utility enterprise typically needed to model the operational aspects of a utility. This model includes public classes and attributes for these objects, as well as the relationships between them.

The objects represented in the CIM are abstract in nature and may be used in a wide variety of applications. The use of the CIM goes far beyond its application in an EMS. This standard should be understood as a tool to enable integration in any domain where a common power system model is needed to facilitate interoperability and plug compatibility between applications and systems independent of any particular implementation.

ENERGY MANAGEMENT SYSTEM APPLICATION PROGRAM INTERFACE (EMS-API) –

Part 301: Common Information Model (CIM) Base

1 Scope

The Common Information Model (CIM) is an abstract model that represents all 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 Energy Management System (EMS) applications developed independently by different vendors, between entire EMS systems developed independently, or between an EMS system and other systems concerned with different aspects of power system operations, such as generation or distribution management. This is accomplished 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 independently of how such information is represented internally.

The object classes represented in the CIM are abstract in nature and may be used in a wide variety of applications. The use of the CIM goes far beyond its application in an EMS. This standard should be understood as a tool to enable integration in any domain where a common power system model is needed to facilitate interoperability and plug compatibility between applications and systems independent of any particular implementation.

Due to the size of the complete CIM, the object classes contained in the CIM are grouped into a number of logical Packages, each of which represents a certain part of the overall power system being modeled. Collections of these Packages are progressed as separate International Standards. This part of IEC 61970 specifies a base set of packages which provide a logical view of the physical aspects of Energy Management System (EMS) information within the electric utility enterprise that is shared between all applications. Other standards specify more specific parts of the model that are needed by only certain applications. Subclause 4.2 below provides the current grouping of packages into standards documents.

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 61850 (all parts), *Communication networks and systems in substations*

ISO 8601, *Data elements and interchange formats - Information interchange - Representation of dates and times*

3 Terms and definitions

For the purposes of this part of IEC 61970, the terms and definitions given in IEC 60050, Annex A of this document and the following apply.

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 **Application Program Interface** **API**

set of public functions provided by an executable application component for use by other executable application components

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

The CIM is partitioned into a set of packages. A package is a general purpose means of grouping related model elements. There is no specific semantic meaning. The packages have been chosen to make the model easier to design, understand and review. The common information model consists of the complete set of packages. Entities may have associations that cross many package boundaries. Each application will use information represented in several packages.

The comprehensive CIM is partitioned into the following packages for convenience, where packages are grouped to be handled as a single standard document as shown:

IEC 61970-301

- Core
- Domain
- Generation
- Generation Dynamics
- LoadModel
- Meas
- Outage
- Production
- Protection
- Topology
- Wires

Future IEC 61970-302

- Energy Scheduling
- Financial
- Reservation

Future IEC 61970-303

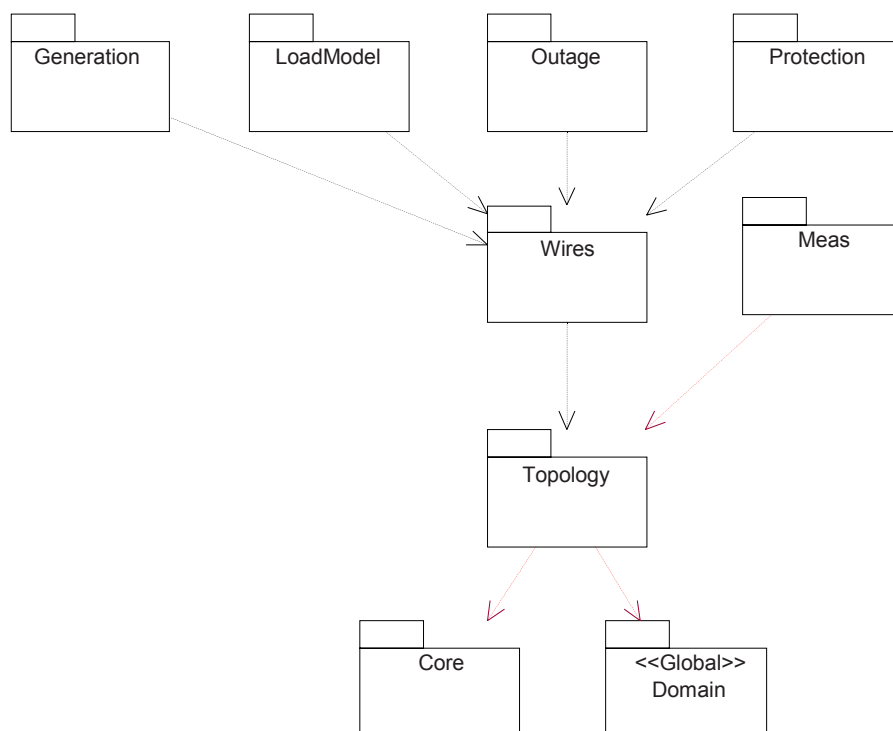
- SCADA

IEC 61968

- Assets
- Consumer
- Core2
- Distribution
- Documentation

Note that the package boundaries do not imply application boundaries. An application may use CIM entities from several packages.

Figure 1 shows the packages defined for IEC 61970-301 CIM Base 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.



IEC 2608/03

Figure 1 – CIM Part 301 Package Diagram

The following Subclauses summarize the contents of each CIM package. Annex A contains the specification for each of the CIM packages.

NOTE 1 The package definitions are loosely based on the “Conformance Blocks” that were defined for the CIM specification version 7 defined in the EPRI CCAP1 project.

NOTE 2 The contents of the CIM defined in this specification were obtained from a straight conversion of the CCAPI CIM static information model defined in the CCAPI CIM Version 10.

NOTE 3 Annex B contains a mapping of the information modeling notation used in the CCAPI CIM Version 7 to the UML used in this standard specification. This Annex is intended to assist those readers who have previously worked with the CCAPI CIM and who now need to adopt the new UML notation. Those readers not acquainted with the previous CCAPI CIM notation may choose to not read Annex B.

4.2.1 Core

This package contains the core Naming, PowerSystemResource, EquipmentContainer, and ConductingEquipment entities shared by all applications plus common collections of those entities. Not all applications require all the Core entities. This package does not depend on any other package, but most of the other packages have associations and generalizations that depend on it.

4.2.2 Topology

This package is an extension to the Core package that in association with the Terminal class models Connectivity, that is the physical definition of how equipment is connected together. In addition, it models Topology, that is the logical definition of how equipment is connected via closed switches. The Topology definition is independent of the other electrical characteristics.

4.2.3 Wires

The Wires package is an extension to the Core and Topology package that models information on the electrical characteristics of Transmission and Distribution networks. This package is used by network applications such as State Estimation, Load Flow and Optimal Power Flow.

4.2.4 Outage

This package is an extension to the Core and Wires packages that model information on the current and planned network configuration. These entities are optional within typical network applications.

4.2.5 Protection

This package is an extension to the Core and Wires packages that model information for protection equipment such as relays. These entities are used within training simulators and distribution network fault location applications.

4.2.6 Meas

The Meas package contains entities that describe dynamic measurement data exchanged between applications.

4.2.7 LoadModel

This package provides models for the energy consumers and the system load as curves and associated curve data. Special circumstances that may affect the load, such as seasons and daytypes, are also included here.

This information is used by Load Forecasting and Load Management.

4.2.8 Generation

The Generation package is divided into two subpackages: Production and Generation Dynamics.

4.2.8.1 Production

This package provides models for various kinds of generators. It also models production costing information which is used to economically allocate demand among committed units and calculate reserve quantities.

This information is used by Unit Commitment and Economic Dispatch of Hydro and Thermal Generating Units, Load Forecasting, and Automatic Generation Control applications.

4.2.8.2 Generation dynamics

This package provides models for prime movers, such as turbines and boilers, which are needed for simulation and educational purposes.

This information is used by the Unit Modeling for Dynamic Training Simulator applications.

4.2.9 Domain

The Domain package is a data dictionary of quantities and units that define datatypes for attributes (properties) that may be used by any class in any other package.

This package contains the definition of primitive datatypes, including units of measure and permissible values. Each datatype contains a value attribute and an optional unit of measure, which is specified as a static variable initialized to the textual description of the unit of measure. Permissible values for enumerations are listed in the documentation for the attribute using UML constraint syntax inside curly braces. String lengths are listed in the documentation and are also specified as a length property.

4.3 CIM classes and relationships

The class diagram(s) for each CIM package shows all the classes in the package and their relationships. Where relationships exist with classes in other packages, those classes are also shown with a note identifying the package which owns the class.

Classes and objects model what is in a power system that needs to be represented in a common way to EMS applications. A class is a description of an object found in the real world, such as a power transformer, generator, or load that needs to be represented as part of the overall power system model in an EMS. Other types of objects include things such as schedules and measurements that EMS applications also need to process, analyze, and store. Such objects need a common representation to achieve the purposes of the EMS-API standard for plug-compatibility and interoperability. A particular object in a power system with a unique identity is modeled as an instance of the class to which it belongs.

It should also be noted that the CIM is defined to facilitate data exchange. As defined in this document, CIM entities have no behavior other than default create, delete, update and read. In order to make the CIM as generic as possible, it is highly desirable to make it easy to configure for specific implementations. In general it is easier to change the value or domain of an attribute than to change a class definition. These principles imply that the CIM should avoid defining too many specific sub-types of classes. Instead the CIM defines generic classes with attributes giving the type name. Applications may then use this information to instantiate specific object types as required. Applications may need additional information to define the set of valid types and relationships.

Classes have attributes that describe the characteristics of the objects. Each class in the CIM contains the attributes that describe and identify a specific instance of the class. Only the attributes that are of public interest to EMS applications are included in the class descriptions.

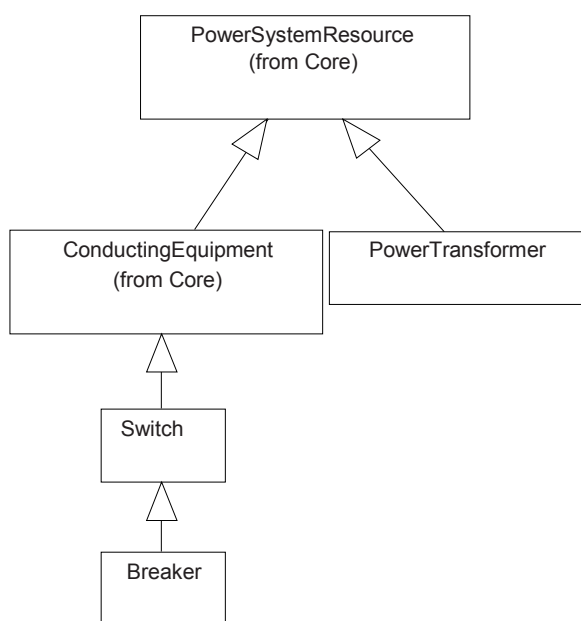
Each attribute has a type, which identifies what kind of attribute it is. Typical attributes are of type integer, float, boolean, string, and enumeration, which are called primitive types. However, many additional types are defined as part of the CIM specification. For example, Compensator has a MaximumkV attribute of type Voltage. The definition of data types is contained in the Domain Package described in Subclause 4.2.9.

Relationships between classes reveal how they are structured in terms of each other. CIM classes are related in a variety of ways, as described in the Subclause below.

4.3.1 Generalization

A generalization is a relationship between a more general and a more specific class. The more specific class can contain only additional information. For example, a Power Transformer is a specific type of Power System Resource. Generalization provides for the specific class to inherit attributes and relationships from all the more general classes above it.

Figure 2 is an example of generalization. In this example, taken from the Wires package, a Breaker is a more specific type of Switch, which in turn is a more specific type of ConductingEquipment, which is itself a more specific type of PowerSystemResource. A PowerTransformer is another more specific type of PowerSystemResource.



IEC 2609/03

Figure 2 – Example of generalization

4.3.2 Simple association

An association is a conceptual connection between classes. Each association has two roles. Each role is a direction on the association that describes the role the target class (i.e., the class the role goes *to*) has in relation to the source class (i.e., the class the role goes *from*). Roles are given the name of the target class with or without a verb phrase. Each role also has multiplicity/cardinality, which is an indication of how many objects may participate in the given relationship. In the CIM, associations are not named.

For example, in the CIM there is an association between a TapChanger and a Regulation Schedule (See Figure 3, which is taken from the Wires package).

Multiplicity is shown at both ends of the association. In this example, a TapChanger object may have 0 or 1 RegulationSchedules, and a RegulationSchedule may belong to 0, 1, or more TapChanger objects.

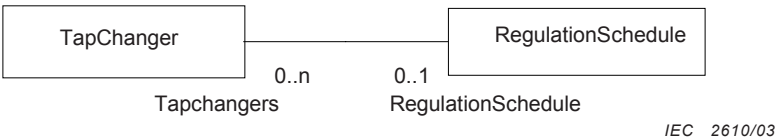


Figure 3 – Example of simple association

4.3.3 Aggregation

Aggregation is a special case of association. Aggregation indicates that the relationship between the classes is some sort of whole-part relationship, where the whole class “consists of” or “contains” the part class, and the part class is “part of” the whole class. The part class does not inherit from the whole class as in generalization.

Figure 4 illustrates an aggregation between the TopologicalIsland class and the TopologicalNode class, which is taken from the Topology package. As shown, a TopologicalNode can be a member of exactly one TopologicalIsland, but a TopologicalIsland can contain any number (but at least one) of TopologicalNodes.

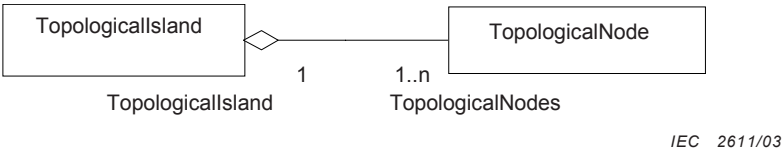


Figure 4 – Example of Aggregation

4.4 CIM model concepts and examples

The CIM classes, attributes, types, and relationships are specified in Annex A. Annex A comprises a complete description of the CIM.

To help understand how to interpret the CIM, four examples are given in the following paragraphs. The first is a portion of the Wires package class diagram illustrating how a PowerTransformer is modeled. The second illustrates how the important concept of Connectivity is modeled in the CIM. The third shows the inheritance hierarchy of devices modeled in the CIM. The fourth illustrates the important concept of equipment containers.

4.4.1 Transformer model

Figure 5 shows a portion of the Wires class diagram, which models a PowerTransformer device.

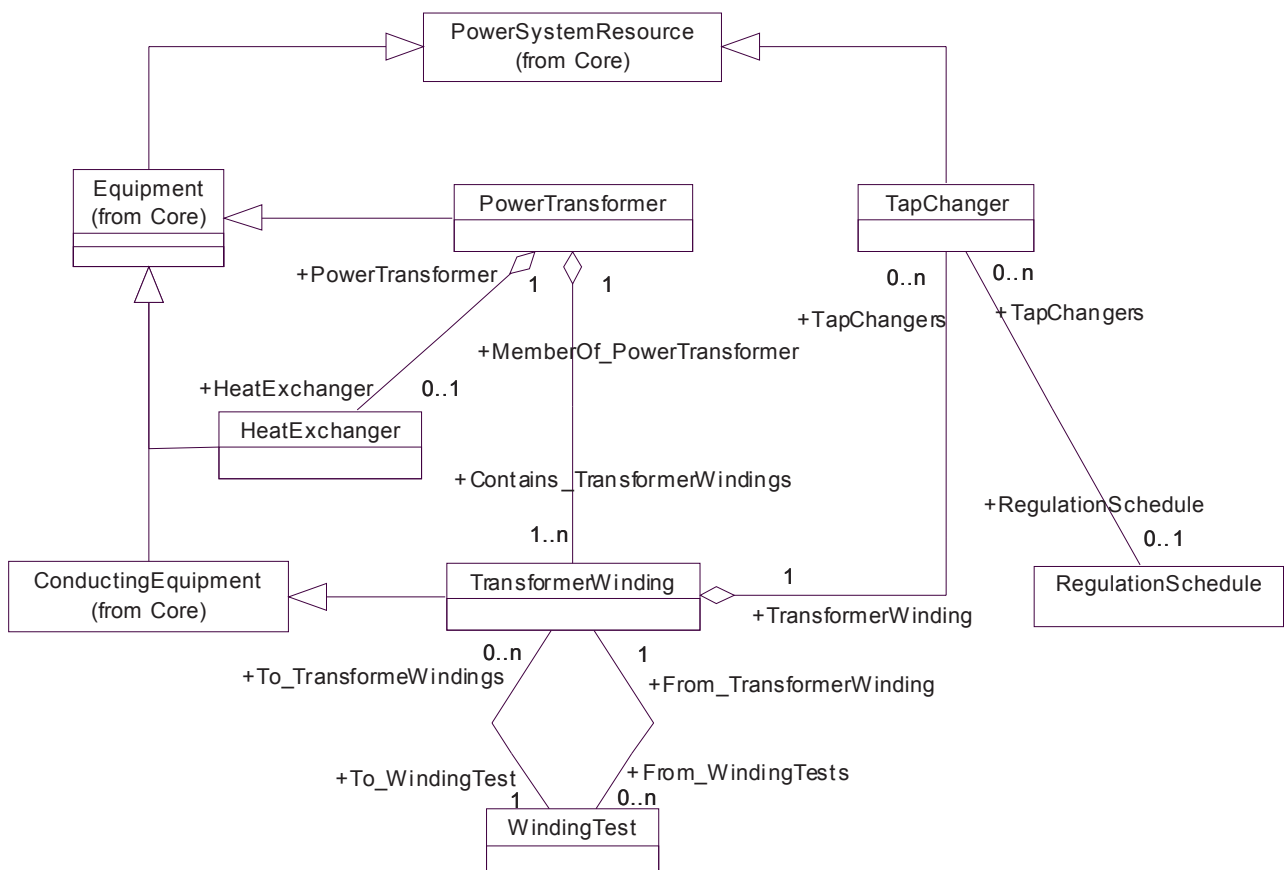
As shown, a PowerTransformer is a specialized class of Equipment, which is a specialized class of a PowerSystemResource, as is ConductingEquipment and TapChanger. This is shown by the use of the generalization-type of relationship, which uses an arrow to point to the general class, and permits the PowerTransformer to inherit attributes from both Equipment and PowerSystemResource.

A PowerTransformer also has a TransformerWinding, which is modeled with an aggregation-type of relationship using a diamond symbol to point from the part class to the whole class. As shown, a PowerTransformer may have (or contain) one or more TransformerWindings, but a TransformerWinding may belong to (or be a member of) only one PowerTransformer.

The TransformerWinding has other relationships as well:

- a generalization relationship with ConductingEquipment;
- an association relationship with the WindingTest class, such that a TransformerWinding object may be TestedFrom from 0, 1, or more WindingTest objects;
- an aggregation relationship with the TapChanger class, such that a TransformerWinding object may have 0, 1, or more TapChanger objects associated with it.

Annex A contains a complete description of each class in Figure 5 along with the definition of all the attributes and relationships supported in each class.



IEC 2612/03

Figure 5 – Transformer model

4.4.2 Connectivity model

Figure 6 shows the Topology class diagram which models connectivity between different types of ConductingEquipment. Also included is a portion of the Meas package class diagram dealing with measurements to illustrate how measurements are associated with conducting equipment.

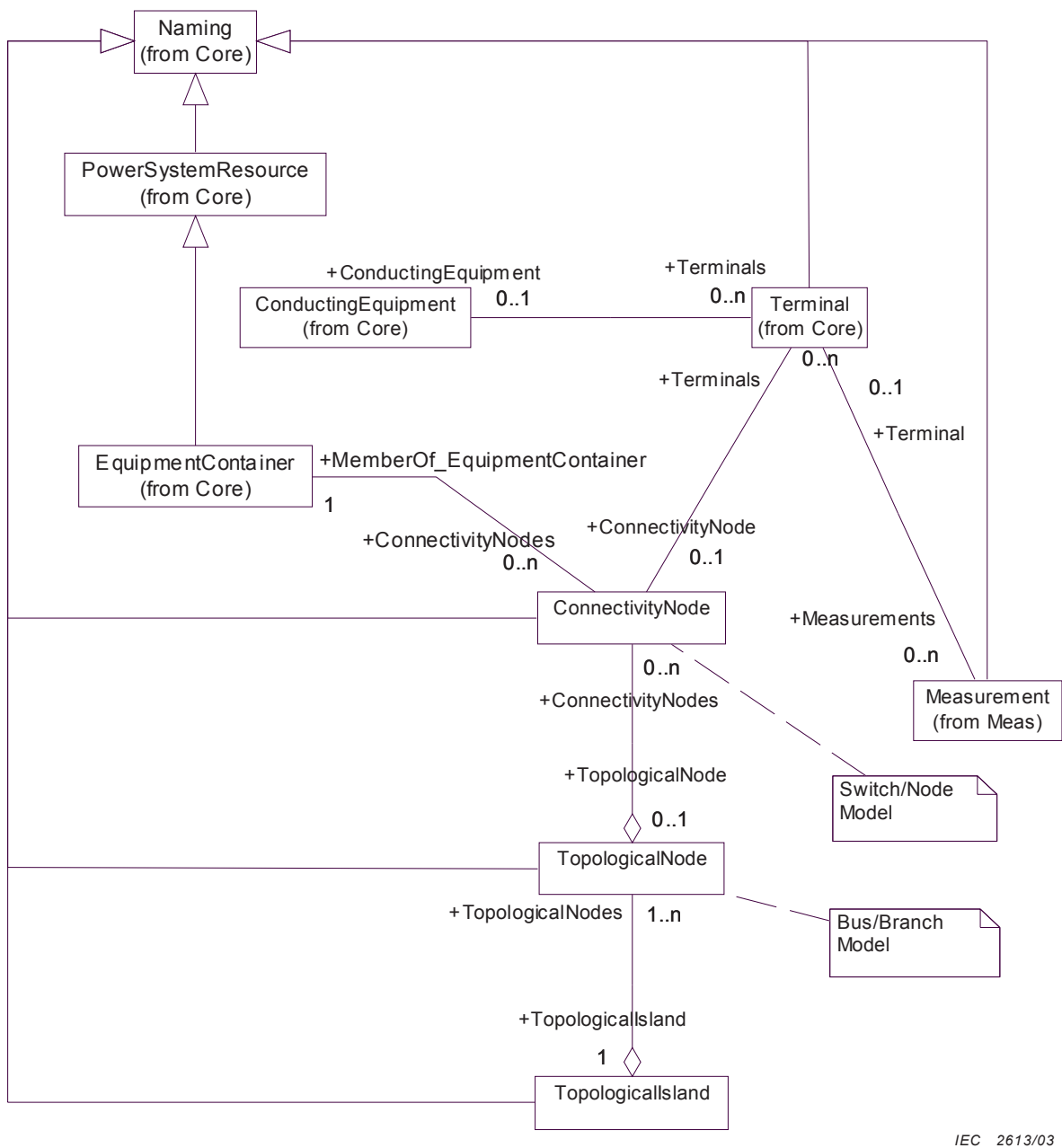


Figure 6 – Connectivity model

To model connectivity, Terminal and Connectivity classes are defined. A Terminal belongs to one ConductingEquipment, although ConductingEquipment may have any number of Terminals. Each Terminal may be connected to a ConnectivityNode, which is a point where terminals of conducting equipment are connected together with zero impedance. A ConnectivityNode may have any number of terminals connected, and may be a member of a TopologicalNode (i.e., a bus), which is in turn a member of a TopologicalIsland. TopologicalNodes and TopologicalIslands are created as a result of a topology processor evaluating the “as built” topology and the actual switch positions.

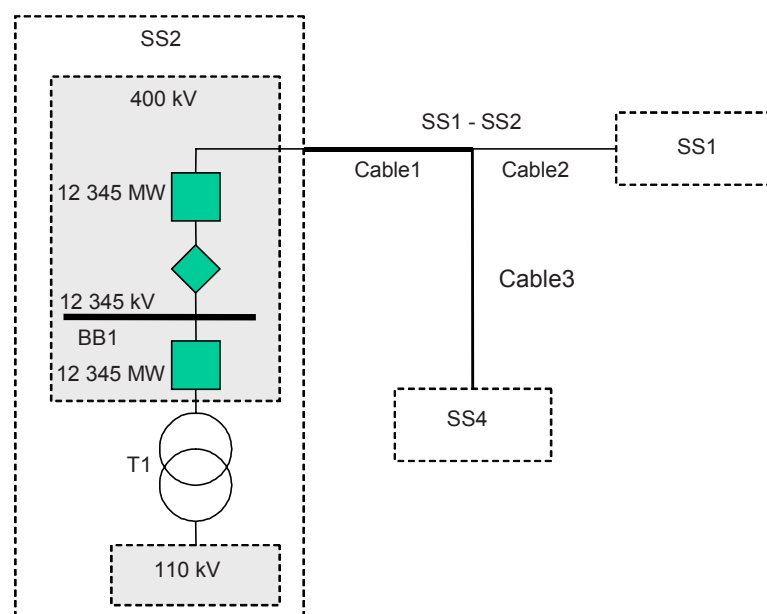
EquipmentContainers, which are a specialization of a PowerSystemResource, may contain zero or more ConnectivityNodes. The associations, ConductingEquipment – Terminal and Terminal – ConnectivityNode, capture the as built topology of an actual power system network. For each Terminal connected to a ConnectivityNode, the associations of the other Terminal(s) connected to the same ConnectivityNode identify the ConductingEquipment object(s) that are electrically connected.

To model the analog values such as voltage and power, each Terminal has an association with a Measurement class from the Meas package. Although not shown in Figure 6, a Measurement object is associated with at least one MeasurementValue object. Each MeasurementValue object is an instance of a measurement from a specific source, for example, a telemetered measurement. In a study context, the measurement values would have a calculation source instead.

Annex A contains a complete description of each class in Figure 6 along with the definition of all the attributes and relationships supported in each class.

4.4.2.1 Connectivity and containment example

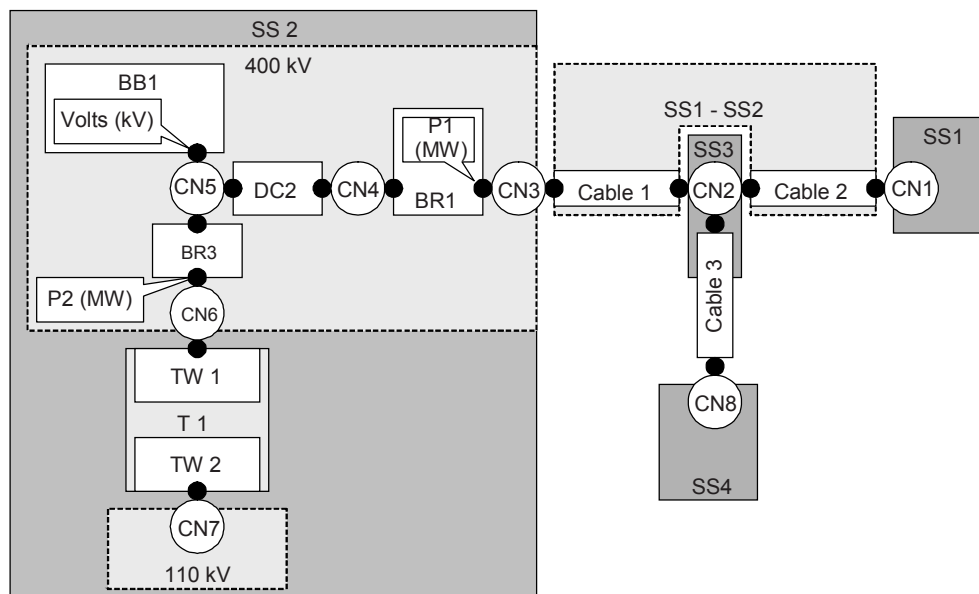
To illustrate how the connectivity model and containment model would appear as objects, a small example is presented in Figure 7. The example shows a transmission line with a T-junction spanning two substations and a substation having two voltage levels with a transformer between them. The transmission line consists of two different cables. One of the voltage levels is shown with a busbar section having a single busbar and two very simple switchgear bays connecting to the busbar.



IEC 2614/03

Figure 7 – Simple Network Example

Figure 8 shows how connectivity is modeled in the CIM as well as one way (but not necessarily the only way) containment is modeled for the diagram in Figure 6. The shaded square boxes represent EquipmentContainers, and the white square boxes represent ConductingEquipment. Darker shading indicates the EquipmentContainer is higher up in the containment hierarchy (i.e., Substation is highest, VoltageLevel next, etc.). White circles represent ConnectivityNodes, and black small circles represent Terminals. A Terminal belongs to a ConductingEquipment, and a ConnectivityNode belongs to an Equipment Container. This means that the borders (or contact points) between ConductingEquipment are their Terminals interconnected via ConnectivityNodes.



IEC 2615/03

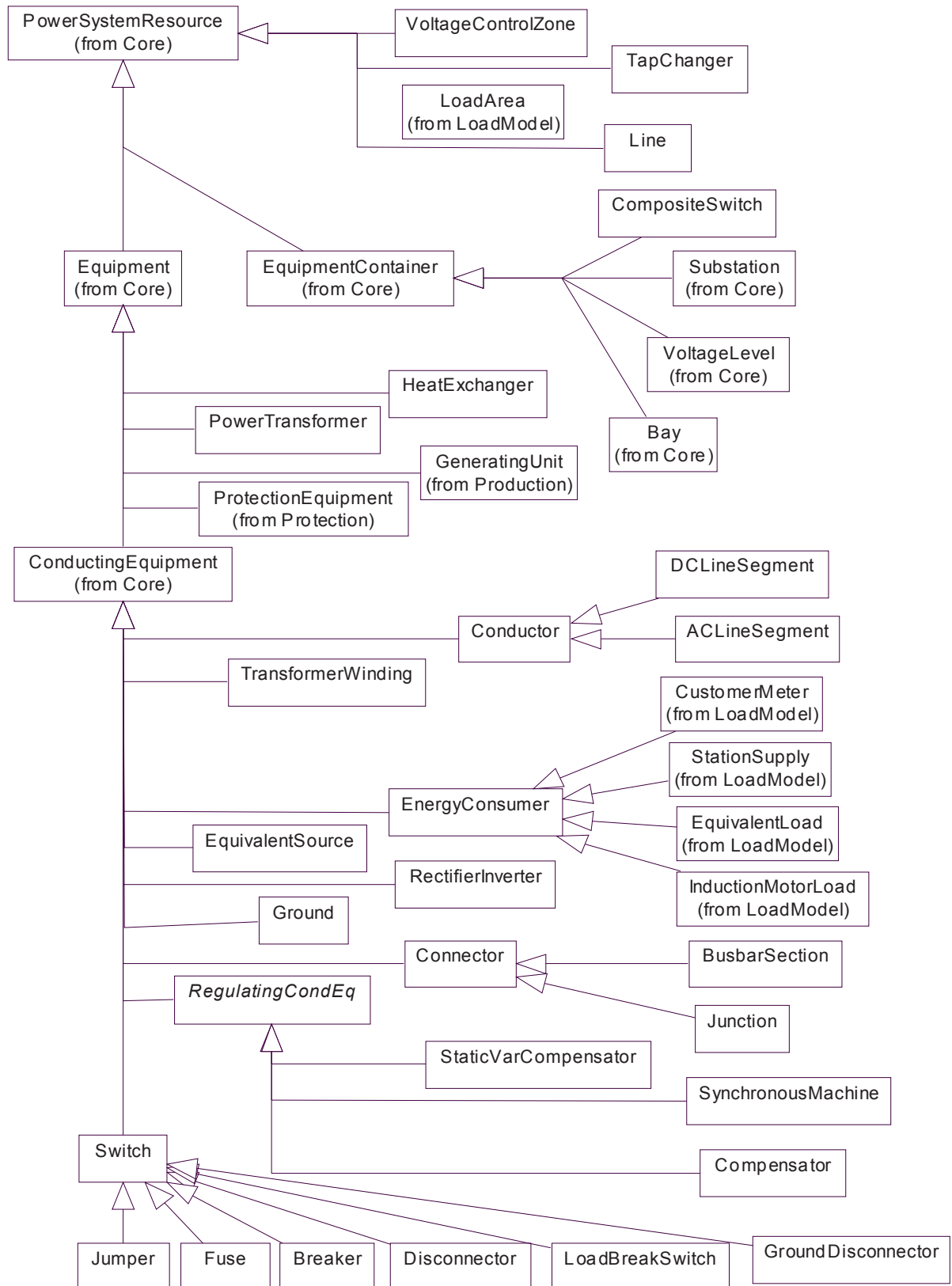
Figure 8– Simple network connectivity modeled with CIM topology

The Line SS1-SS2 has two ACLineSegments – Cable1 and Cable2. A collapsed Substation SS3 with ConnectivityNode CN2 models the connection point between the ACLineSegments as well as a T-junction to Cable3, which provides a connection to Substation 4. Each ACLineSegment has two Terminals. Cable1 is connected to CN3 and CN2 via these Terminals. CN3 is contained by the VoltageLevel 400 kV. The breaker BR1 has two terminals of which one is connected to CN3. The rest is left to the interested reader to trace.

Measurements are represented by square callouts where the arrow points to a Terminal. P1 is connected to the right Terminal belonging to Breaker BR1. Note that P1 is drawn inside the box representing BR1. This is because a Measurement may belong to a Power System Resource (PSR), as is the case with BR1. P2 is drawn inside the VoltageLevel 400 kV, which means it belongs to the 400 kV VoltageLevel instead of BR3.

4.4.3 Inheritance Hierarchy

Figure 9 shows an overview of the inheritance hierarchy modeled in the CIM. This overview, which is included as one of the Wires package diagrams, actually spans most of the CIM packages.

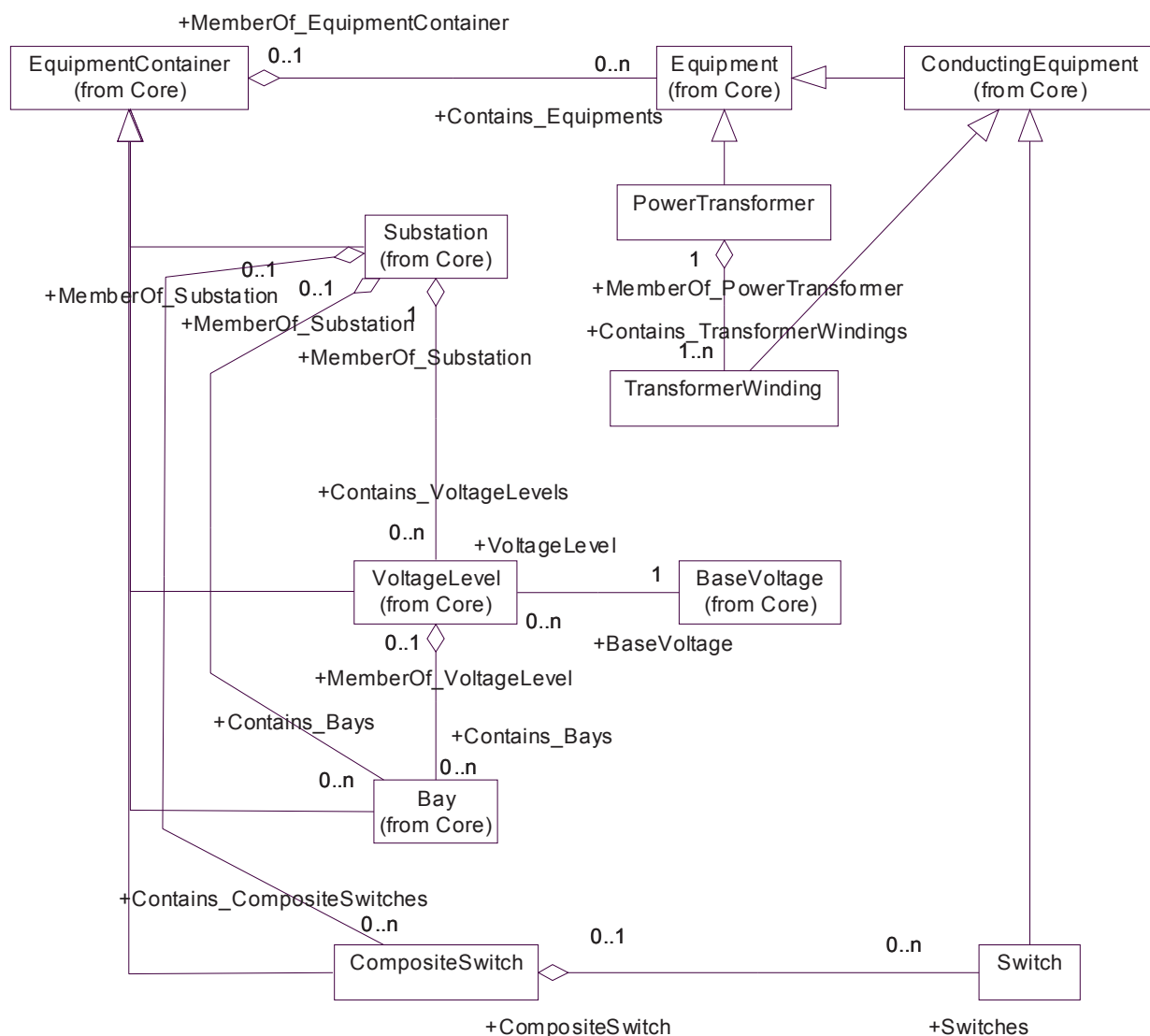


IEC 2616/03

Figure 9– Equipment inheritance hierarchy

4.4.4 Equipment Containers

Figure 10 shows the concept of equipment containers modeled in the CIM. Equipment containers represent ways of organizing and naming equipment typically found within a substation. As may be seen, there is some flexibility provided in which containers are used in a specific application of the CIM in order to accommodate different international practices as well as differences typically found between transmission and distribution substations. Each container represents an aggregation of other containers and/or equipment objects.



IEC 2617/03

Figure 10 – Equipment containers

4.5 Modeling tools

The CIM was constructed and is maintained using Rational Rose from Rational Software Corporation. The entire CIM exists as a .mdl file viewable with Rational Rose, including the class diagrams and descriptions of classes, attributes, types, and relationships. Viewing the CIM in this fashion provides a graphical navigation interface that permits all CIM specification data to be viewed via point-and-click from the class diagram in each package. Each top level package is also distributed as a .cat file, allowing new models to be constructed from the CIM packages.

All future changes to the CIM specification, resulting in new versions of this standard, will be incorporated first into the Rational Rose model description to ensure a single source for the CIM model data.

Annex A of this document was auto-generated using Rational SoDA, another modeling tool from Rational Software Corporation. SoDA generates Word documents from the Rose model file based on a custom template created in Word that incorporates the IEC format and styles.

4.6 Modeling guidelines

This section provides guidelines on how to maintain and extend the CIM.

The CIM is meant to contain classes and attributes that will be exchanged over public interfaces between major applications. The goal is to keep, as much as possible, only the generic features from which a detailed implementation may be derived. In general, it is easier to change the value or domain of an attribute than to change a class definition. This makes the model more robust because it is able to support a broader class of requirements, and more stable because new requirements may be able to be handled without requiring changes to the model.

4.6.1 Amendments to the CIM

From time to time it may be desirable to amend the CIM to either revise the existing model or to extend the CIM to model additional elements of an electric utility power system. The recommended process for such amendments is as follows:

- a) Prepare a Use Case(s) to describe the desired changes. This should include proposed changes to the appropriate class diagrams showing new/revised classes, attributes, and associations.
- b) The Use Case(s) is then reviewed by the appropriate IEC Working Group to decide if the requested changes should be treated as revisions to the current CIM standard, or if they should be treated as private amendments, not requiring a change to the standard itself.
- c) Proposed amendments accepted by the working group will be added to a list of outstanding issues, and at the appropriate time, a new version of the CIM model will be prepared and an update made to the appropriate IEC CIM specification.

4.6.1.1 Changes to the CIM UML Rose model

From a modeling perspective, when the CIM is extended, the approach is to start with the existing CIM UML model in Rational Rose format. The extensions may be added in any of several ways that are available in UML, but in all cases the approach is to inspect the current model and determine the best way to build off of the existing class diagrams. The extensions may take the form of any of the following, starting from the simplest to the most complex:

- adding additional values to existing attributes in a class;
- adding additional attributes to existing classes;
- adding new classes that are specializations of existing classes;
- adding new classes via associations with existing classes.

The main objective is to reuse the existing CIM to the maximum extent possible. From a packaging point of view, extensions should be made to existing Packages where possible. If the extensions comprise a new domain of application, then consideration should be given to creating a new Package for the additions, but still creating the necessary associations to the existing package, keeping in mind that even though a new Package is being created, the CIM is still a single Rose model file.

4.6.1.2 Changes to the CIM standards documents

From a documentation perspective, when the CIM is extended, a decision must be made whether the changes constitute updates to existing CIM standards documents, or whether a new IEC 61970-3xx specification is required. In either case, the extensions will then become part of the IEC standard CIM.

4.6.2 CIM profiles

An implementation of the CIM need not include all classes, attributes, or associations in the standard CIM specification to be compliant with the CIM standard. Profiles may be defined to specify which elements must be included (i.e., mandatory elements) in a particular use of the CIM, as well as which are optional. These profiles will be defined in the IEC 61970-4xx series of standards.

An example is the profile for exchanging power system models, which is the subject of the Future IEC 61970-452 CIM Model Exchange Specification. This document specifies how the CIM is to be used for exchanging power system models in XML, and also specifies the mandatory and optional classes, attributes, and associations to be supported for this use of the CIM as required by NERC (North American Electric Reliability Council). The CIM version 10 Rose model includes this profile information in the form of a special tab labeled "CIM" in the specification for each class, attribute, and association to indicate if the item is included or not in this profile. Under this tab for a given item is a list of the profiles currently defined and the value assigned to that item (i.e., included or not in the identified profile). For example, in the Measurement class, there is one entry "NercProfile" with the value "True", indicating that this class is mandatory for exchanging power system models that are based on the NERC Profile specification. As new profiles are defined, entries can be made for them as well.

4.7 User implementation conventions

This subclause provides recommended user conventions when using the CIM in actual system implementations.

4.7.1 Naming

The Naming class contained in the Core package is inherited by all PowerSystemResource classes. This class contains four attributes to be used for naming all PowerSystemResource objects. It is intended that values should be assigned to each attribute in a manner consistent with the attribute definitions.

The following are definitions and conventions recommended for naming attributes of PowerSystemResource classes:

- name: A local short name of the instance. Objects that are structured in a containment hierarchy have this name local to each level in the hierarchy. The name must be unique among objects contained by the same parent.
- pathName: Objects that are structured in a containment hierarchy have a pathName. The pathName is a concatenation of all names from each container, i.e., from the leaf object up to the root of the containment hierarchy, similar to a file path name. For example, if node "A" contains node "B" that contains node "C", then the pathName for node "C" may look like "A.B.C". The type of delimiters used between names is not specified but is a local implementation issue.
- aliasName: A free text name of the instance. This attribute can also be used for localization.
- description: A free format description of the instance.

The following conventions are recommended for naming attributes for non-PowerSystem Resource classes:

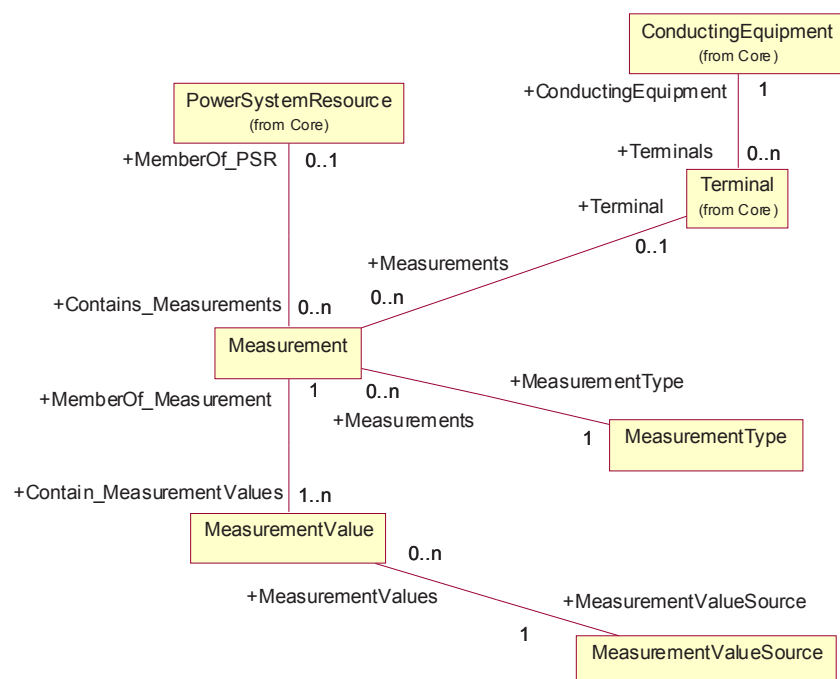
- name: Main classification of the instances. Example: Unit name for the units class, measurement type name for the MeasurementType class.
- pathName: Optional second level classification of the instances.
- aliasName: Abbreviations.
- description: A free format description of the instance.

4.7.2 Use of Measurement-related classes

A PowerSystemResource (PSR) may have zero to many measurements associated with it, either by containing a measurement directly or through one of its terminals. Each measurement may have one or more measurement values. Observing the following guidelines will enable applications to navigate and find the required measurement values in a consistent way (see Figure 11):

- a) Measurements of a PowerSystemResource are classified by MeasurementType.
- b) MeasurementValues of a Measurement are classified by MeasurementValueSource.
- c) MeasurementType¹ inherits from Naming. The values to be used for Measurement Type.name, MeasurementType.aliasName, and MeasurementType.description are specified in Table 1, where:
 - MeasurementType.name is the IEC 61970 name assigned;
 - MeasurementType.aliasName is the name assigned to the type in the IEC 61850 series;
 - MeasurementType.description defines the type.
- d) MeasurementValueSource also inherits from Naming. The values to be used for MeasurementValueSource.name and MeasurementValueSource.description are given in Table 2. This table provides a number of source names to be used where possible. However, the exact names to be used for specific applications are defined in related IEC 61970 Component Interface Specifications (CIS).
- e) The tables may be extended for proprietary needs. The names added shall start with a unique name (for example the company name) and an underscore. Example: xyz_Average Temperature.

¹ MeasurementType is different from Unit, which is part of the Domain package described in 4.2.9. MeasurementType describes "what" is measured rather than the unit of measure including scaling, which is the purpose of Unit.



IEC 2618/03

Figure 11 – Navigating from PSR to MeasurementValue

Table 1 – MeasurementType naming conventions

Name	aliasName	Description
Current	Amps	Current I (rms) of a non-three phase circuit
ThreePhaseCurrent	TotAmps	Total current I (r.m.s.) in a three phase circuit
Frequency	Hz	Frequency (f)
PowerFactor	PwrFact	Power Factor (pf)
ThreePhasePowerFactor	TotPF	Average power factor (pf) in a three phase circuit
ThreePhaseApparentPower	TotVA	Total apparent power (S) in a three phase circuit
ThreePhaseReactivePower	TotVAR	Total reactive power (Q) in a three phase circuit
ThreePhaseActivePower	TotW	Total real power (P) in a three phase circuit
ApparentPower	VoltAmp	Apparent power (S) in a non-three phase circuit
ReactivePower	VoltAmpR	Reactive power (Q) in a non-three phase circuit
Voltage	Volts	Voltage (V) (r.m.s.)
ActivePower	Watts	Real power (P) in a non-three phase circuit
Pressure	Pres	Pressure
Temperature	Temp	Temperature
ThreePhaseAngle	TotAng	Angle (ϕ) [in a three phase circuit]
ApparentEnergy	TotVAh	Apparent energy
ReactiveEnergy	TotVARh	Reactive energy
ActiveEnergy	TotWh	Real energy
Automatic	Auto	Automatic operation (not manual). automatic = TRUE
LocalOperation	Loc	Local operation (not remote). local = TRUE
AutomaticControl	LTCBlk	Automatic control of LTC blocked (inhibited). blocked = TRUE
SwitchPosition	Pos	Switch position (2 bits = intermediate, open, closed, ignore)
TapPosition	TapPos	Tap position of power transformer or phaseshifter
Operation Count	OperCnt	Operation count - typically for switches
ParallelTransformers	Parallel	Transformers in parallel operating mode

Table 2 – MeasurementValueSource naming conventions

Name	Description
SCADA	Telemetered values received from a local SCADA system
CCLink	Value received from a remote control center via TASE.2 or other control center protocol
Operator	Operator entered value (always manually maintained, PSR is not connected to an RTU)
Estimated	Value updated by a state estimator
PowerFlow	Value updated as result of a Powerflow
Calculated	Calculated from other measurement values (for example, a sum)
Allocated	Calculated by a load allocator

Following these conventions:

- Each Measurement instance represents a technological quantity of a Power System Resource.
- Each MeasurementValue of a Measurement represents a current value for the technological quantity, as supplied from a single source.
- The *source* attribute in MeasurementValueQuality then indicates whether the source actually provided the current value, or whether it had been substituted or defaulted.

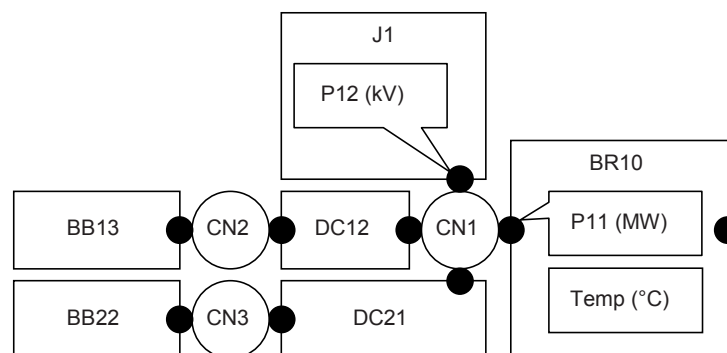
Attachment of Measurements

As mentioned in 4.7.2 and as shown in Figure 11, Measurements can be attached to equipment in two ways:

- 1) contained by a PowerSystemResource;
- 2) via a Terminal part of a ConductingEquipment.

The first option is used for Measurements that are not related to connectivity, for example temperature, weight, size.

The second option is used for connectivity-related Measurements, for example power flows, voltages, currents. Voltages have no direction and can be attached wherever appropriate in relation to the sensor placement. Flows have direction and must be attached such the flow direction is evident from the placement. Figure 12 shows two examples of the placement of Measurements.



IEC 2619/03

Figure 12 – Measurement placement

P12 is a voltage Measurement that measures the voltage at the Junction J1. P12 is topologically related to the ConnectivityNode CN1 via the Terminal in Junction J1. P11 is a Measurement that measures the flow through Breaker BR10 at the side connected to the ConnectivityNode CN1. P11 is topologically related to the ConnectivityNode CN1 via the left Terminal in Breaker BR10. Temp is a Measurement that measures the Breaker temperature.

As a temperature is not related to connectivity, it has no relation to a Terminal – it just belongs to the Breaker BR10.

4.7.3 Number of Terminals for ConductingEquipment Objects

The following ConductingEquipment classes have two terminals: ACLineSegment, DCLineSegment, Jumper, Fuse, Breaker, Disconnecter, LoadBreakSwitch. All other ConductingEquipment classes have a single terminal, except for the Compensator class, which has one terminal if the attribute compensatorType has value “shunt”, and two terminals if the attribute compensatorType has value “series”.

For instances of ConductingEquipment that are not electrically connected to other equipment (i.e., as when treated as an Asset in inventory), it is acceptable to have no terminals specified.

4.8 Examples

Power system models have been created from the CIM UML model in various ways. The first example is an RDF (Resource Description Framework) Schema version of the CIM, which uses XML (eXtensible Markup Language) to describe the power system model (see Bibliography, References 1 to 3). RDF Schema versions of the CIM have been used to create XML model files of actual networks for purposes of interoperability testing. An RDF Schema version of the CIM is generated from the CIM UML Rose model file using a software tool called Xpetal based on the RDF Schema specification of the CIM. Xpetal is available free of charge on the Web from SourceForge.

It should be noted that an RDF schema version of the CIM is still metadata rather than an instantiation of an actual network. However, complete network model files with descriptions of all network elements and their electrical connectivity can be generated by system suppliers using proprietary export tools, and then imported by other systems via a similar import tool, which is used to populate a local network engineering tool database. Examples of such CIM XML models include the Siemens 100 bus model, the Alstom 60 bus model, and the ABB 40 bus model files used for CIM XML interoperability testing (see Bibliography, References 4 to 7 for published reports of these recent interoperability tests and the model files used).

Annex A (normative)

Common information model for control center application program interface

A.1 Logical architecture

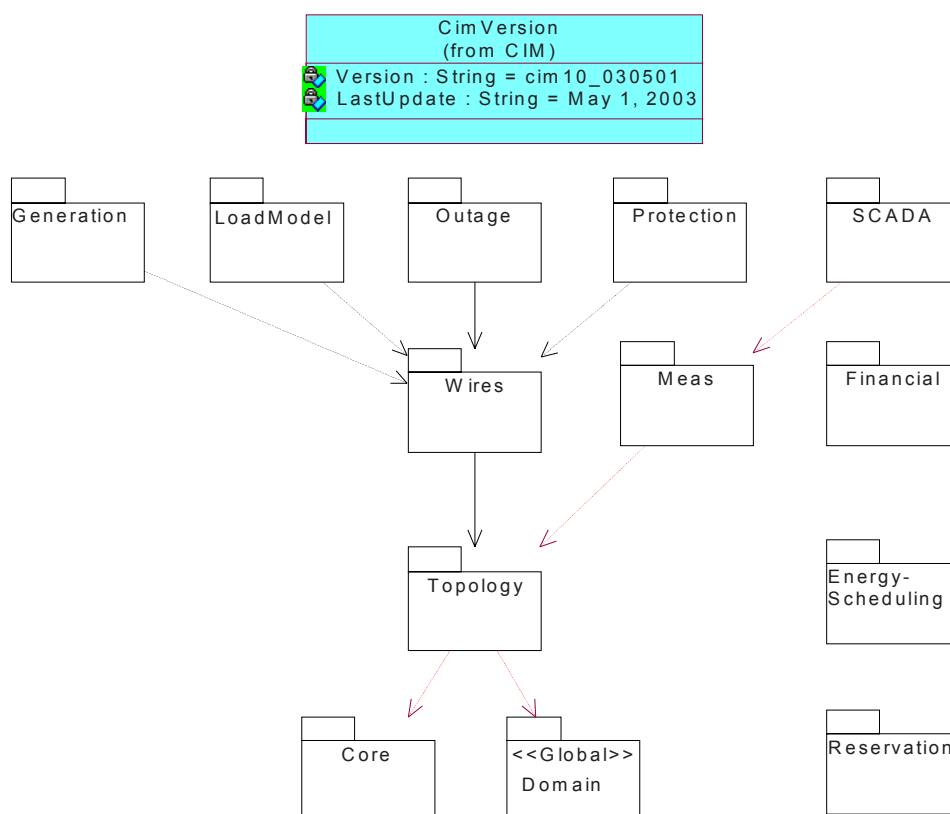
A.1.1 Overview

The Common Information Model (CIM) represents a comprehensive logical view of energy management system information. This definition includes the public classes and attributes, as well as the relationships between them.

A.1.2 Context

The CIM is partitioned into subpackages. The Domain package defines datatypes used by the other packages. The Generation package is subdivided into Production and Generation Dynamics packages. 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. Aggregations are listed only for the role that contains the aggregation.

Figure A.1 shows the top level packages and the dependencies between them.



IEC 2620/03

Figure A.1 – CIM top level packages

ParentClass.Name	(Type)	Documentation
------------------	--------	---------------

Note that the `ParentClass` is omitted for the native attributes.

The classes in the Domain package include an optional unit of measure for the attribute type.

The associations are listed according to the role the class participates in for that association. The role information for native and inherited roles is listed as:

(MultiplicityFrom)	<i>RoleTo.Name</i>	(MultiplicityTo)	<u>Role.ToClass. Name</u>	Association Documentation
--------------------	--------------------	------------------	-------------------------------	------------------------------

The MultiplicityFrom indicates the multiplicity from the class that is being described. A value of zero indicates an optional association. A value of n means that an unspecified number of associations are permitted. The RoleTo.Name is the role of the target class on the other side of the association. The MultiplicityTo and Role.ToClass.Name indicate the multiplicity and name of the class on the other side of the association.

A.2 Package Architecture

This specification is autogenerated from the CIM model file cim10_030501.mdl

A.2.1 CIM

The Common Information Model (CIM) represents a common logical view of EMS public data. The CIM definition consists of a comprehensive information model. This information model is largely independent of any specific technological implementation methods.

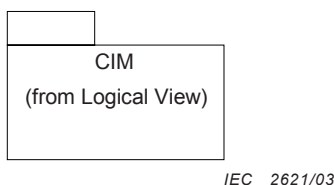


Figure A.2 – Main

A.2.1.1 CimVersion

This is the CIM version number assigned to this UML model file. The version naming convention is to use the last major revision file name followed by the date of the last revision (for example, cimv08b 001212).

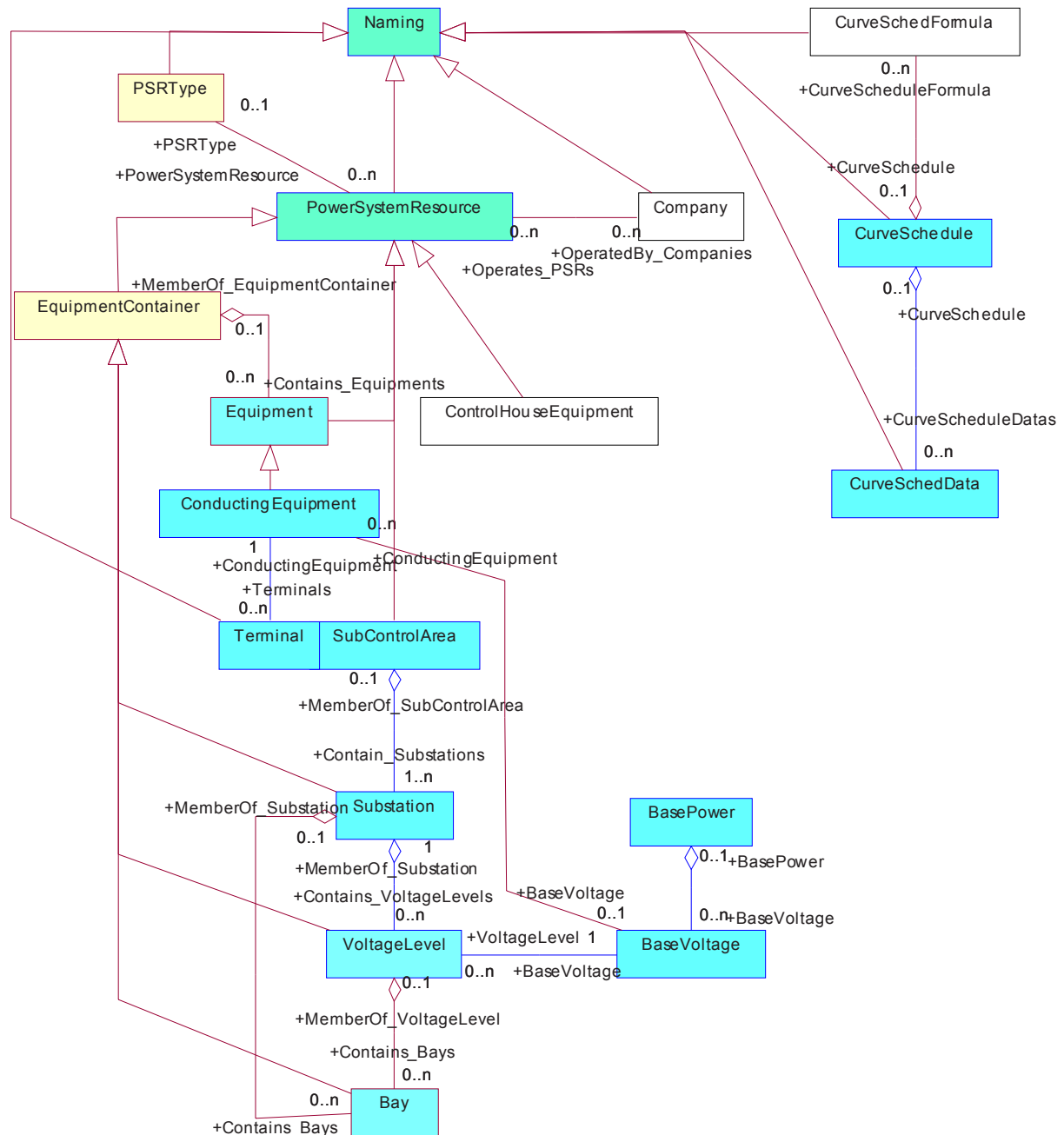
CimVersion Attributes

Native Attributes

Version	(String)
LastUpdate	(String)

A.2.2 Core

Contains the core PowerSystemResource and ConductingEquipment entities shared by all applications plus common collections of those entities. Not all applications require all the Core entities.



IEC 2622/03

Figure A.3 – Main

Figure A.3 is intended to show all classes needed for any application of the CIM for modeling transmission and generation systems.

A.2.2.1 BasePower

The BasePower class defines the base power and base voltages that are used in the per unit calculations.

BasePower Attributes*Native Attributes*

basePower (ApparentPower) Definition of base power.

BasePower Associations*Native Roles*

(0..1) BaseVoltage (0..n) BaseVoltage Used to specify that voltages in the CIM are given a value existing in the collection.

A.2.2.2 BaseVoltage

Collection of BaseVoltages which is used to verify that the BusbarSection.BaseVoltage and other voltage attributes in the CIM are given a value existing in the collection.

BaseVoltage Attributes*Native Attributes*

nominalVoltage (Voltage) The PowerSystemResource's base voltage.

BaseVoltage Associations*Native Roles*

(0..1) ConductingEquipment (0..n) Conducting Equipment Use association to Conducting Equipment only when there is no VoltageLevel container used.

(1) VoltageLevel (0..n) VoltageLevel

A.2.2.3 Bay

A collection of power system resources (within a given substation) including conducting equipment, protection relays, measurements, and telemetry.

Bay Attributes*Native Attributes*

bayEnergyMeasFlag (Boolean) Indicates the presence/absence of kWh/kvarh measurements.

bayPowerMeasFlag (Boolean) Indicates the presence/absence of MW/Mvar measurements.

breakerConfiguration (BreakerConfiguration) Breaker configuration.

busBarConfiguration (BusbarConfiguration) Bus bar configuration.

Inherited Attributes

Naming.aliasName (String) Free text name of the object or instance.

Naming.description (String) Description of the object or instance.

Naming.name (String) Unique name among objects owned by the same parent.

Naming.pathName (String) pathName is a concatenation of all names from each container.

A.2.2.4 Company

A company is a legal entity that owns and operates power system resources and is a party to interchange and transmission contracts.

Company Attributes

Native Attributes

companyType	(CompanyType)	The type of company, for example: pool, municipal, private.
-------------	---------------	---

Inherited Attributes

Naming.aliasName	(String)	Free text name of the object or instance.
Naming.description	(String)	Description of the object or instance.
Naming.name	(String)	Unique name among objects owned by the same parent.
Naming.pathName	(String)	pathName is a concatenation of all names from each container.

Company Associations

Native Roles

(0..n)	<i>Operates_PSRs</i>	(0..n)	<u>PowerSystemResource</u>	A power system resource may be part of one or more companies.
(0..n)	<i>ContractOrTariffs</i>	(0..n)	<u>ContractOrTariff</u>	A ServiceAgreement is a Custom Instance of a legal agreement between a Company and a ContractOrTariff.
(1..1)	<i>BusinessUnits</i>	(1..n)	<u>BusinessUnit</u>	A Company is organized into one or more BusinessUnits.

A.2.2.5 ConductingEquipment

The parts of the power system that are designed to carry current or that are conductively connected therewith. ConductingEquipment is contained within an EquipmentContainer that may be a Substation, or a VoltageLevel or a Bay within a Substation.

ConductingEquipment Attributes

Native Attributes

phases	(PhaseCode)	Describes the phases carried by conducting equipment. Possible values {ABCN, ABC, ABN, ACN, BCN, AB, AC, BC, AN, BN, CN, A, B, C, N}.
--------	-------------	---

Inherited Attributes

Naming.aliasName	(String)	Free text name of the object or instance.
Naming.description	(String)	Description of the object or instance.
Naming.name	(String)	Unique name among objects owned by the same parent.
Naming.pathName	(String)	pathName is a concatenation of all names from each container.

ConductingEquipment Associations

Native Roles

(1)	<i>Terminals</i>	(0..n)	<u>Terminal</u>	ConductingEquipment has 1 or 2 terminals that may be connected to other ConductingEquipment terminals via ConnectivityNodes.
-----	------------------	--------	-----------------	--

(0..n)	BaseVoltage	(0..1)	<u>BaseVoltage</u>	Use association to ConductingEquipment only when there is no VoltageLevel container used.
(1)	ClearanceTags	(0..n)	<u>ClearanceTag</u>	Conducting equipment may have multiple clearance tags for authorized field work.
(0..n)	ProtectionEquipments	(0..n)	<u>ProtectionEquipment</u>	Protection equipment may be used to protect specific Conducting Equipment. Multiple equipment may be protected or monitored by multiple protection equipment.
<i>Roles Inherited From PowerSystemResource</i>				
(0..n)	OperatedBy_Companies	(0..n)	<u>Company</u>	A power system resource may be part of one or more companies.
(0..n)	PSRType	(0..1)	<u>PSRType</u>	
(0..1)	Contains_Measurements	(0..n)	<u>Measurement</u>	Measurement-PSR defines the measurements in the naming hierarchy.
(1)	OutageSchedule	(0..1)	<u>OutageSchedule</u>	A power system resource may have an outage schedule.
(0..n)	BusinessUnit	(0..1)	<u>BusinessUnit</u>	A power system resource may be operated by one BusinessUnit at a time.

A.2.2.6 ControlHouseEquipment

Equipment in a substation control house. Covers things such as fire alarms, ambient temperature, door alarms, and spares.

ControlHouseEquipment Attributes

<i>Native Attributes</i>		
controlHouseEquipType	(ControlHouseEquipmentType)	Type of control house equipment.
<i>Inherited Attributes</i>		
Naming.aliasName	(String)	Free text name of the object or instance.
Naming.description	(String)	Description of the object or instance.
Naming.name	(String)	Unique name among objects owned by the same parent.
Naming.pathName	(String)	pathName is a concatenation of all names from each container.

A.2.2.7 CurveSchedData

Data point values for defining a curve or schedule

CurveSchedData Attributes

<i>Native Attributes</i>		
rampData	(Float)	The data value of the rate-of-change of the Y-axis variable with respect to the X-axis variable.
rampDataValue	(Float)	The data value of the rate-of-change of the Y-axis variable with respect to the X-axis variable (0 = instantaneous change).
xAxisData	(Float)	The data value of the X-axis variable, depending on the X-axis units.

y1AxisData	(Float)	The data value of the first Y-axis variable, depending on the Y-axis units.
y2AxisData	(Float)	The data value of the second Y-axis variable (if present), depending on the Y-axis units.

Inherited Attributes

Naming.aliasName	(String)	Free text name of the object or instance.
Naming.description	(String)	Description of the object or instance.
Naming.name	(String)	Unique name among objects owned by the same parent.
Naming.pathName	(String)	pathName is a concatenation of all names from each container.

A.2.2.8 CurveSchedFormula

Mathematical formula for defining a curve or schedule segment

CurveSchedFormula Attributes*Native Attributes*

xLowerBound	(Float)	The minimum value of the X-variable for the range of the function.
xUpperBound	(Float)	The maximum value of the X-variable for the range of the function.
yFunction	(Description)	Mathematical formula for the Y-axis variable with numerical constants.

Inherited Attributes

Naming.aliasName	(String)	Free text name of the object or instance.
Naming.description	(String)	Description of the object or instance.
Naming.name	(String)	Unique name among objects owned by the same parent.
Naming.pathName	(String)	pathName is a concatenation of all names from each container.

A.2.2.9 CurveSchedule

Relationship between an independent variable (X-axis) and one or two dependent variables (Y1-axis and Y2-axis). Curves can also serve as schedules.

CurveSchedule Attributes*Native Attributes*

curveStyle	(CurveStyle)	The style or shape of the curve.
rampMethod	(RampMethod)	The deltaY versus deltaX units of measure. Applies to all ramps.
rampStartMethod	(RampStartMethod)	The method of applying the ramp: 0 at start point, 50 % at start point, 100 % at start point. For methods 2 and 3, the ramp begins ahead of the start point on the X-axis. NOTE For storage, all ramps are to be normalized to Method "1" (0 at start point).
rampUnits	(RampUnits)	The deltaY versus deltaX units of measure. Same for "two" Y values.
xAxisType	(XAxisType)	The type of independent variable.
xAxisUnits	(XAxisUnits)	The X-axis units of measure.
y1AxisUnits	(Y1AxisUnits)	The Y1-axis units of measure.
y2AxisUnits	(Y2AxisUnits)	The Y2-axis units of measure.
yAxisType	(YAxisType)	The type of dependent variable.

Inherited Attributes

Naming.aliasName	(String)	Free text name of the object or instance.
Naming.description	(String)	Description of the object or instance.
Naming.name	(String)	Unique name among objects owned by the same parent.
Naming.pathName	(String)	pathName is a concatenation of all names from each container.

CurveSchedule Associations

Native Roles

(0..1)	CurveScheduleDatas	(0..n)	<u>CurveSchedDa</u> <u>ta</u>	The point data values that define a curve.
(0..1)	CurveSchedule Formula	(0..n)	<u>CurveSchedFo</u> <u>rmula</u>	The formula for a curve segment.

A.2.2.10 Equipment

The parts of a power system that are physical devices, electronic or mechanical.

Equipment Attributes

Inherited Attributes

Naming.aliasName	(String)	Free text name of the object or instance.
Naming.description	(String)	Description of the object or instance.
Naming.name	(String)	Unique name among objects owned by the same parent.
Naming.pathName	(String)	pathName is a concatenation of all names from each container.

A.2.2.11 EquipmentContainer

A modeling construct to provide a root class for all Equipment classes.

EquipmentContainer Attributes

Inherited Attributes

Naming.aliasName	(String)	Free text name of the object or instance.
Naming.description	(String)	Description of the object or instance.
Naming.name	(String)	Unique name among objects owned by the same parent.
Naming.pathName	(String)	pathName is a concatenation of all names from each container.

EquipmentContainer Associations

Native Roles

(1)	ConnectivityNodes	(0..n)	<u>ConnectivityNode</u>
(0..1)	Contains_Equipments	(0..n)	<u>Equipment</u>

Roles Inherited From PowerSystemResource

(0..n)	OperatedBy_ Companies	(0..n)	<u>Company</u>	A power system resource may be part of one or more companies.
(0..n)	PSRType	(0..1)	<u>PSRType</u>	
(0..1)	Contains_Measure ments	(0..n)	<u>Measurement</u>	Measurement-PSR defines the measurements in the naming hierarchy.
(1)	OutageSchedule	(0..1)	<u>OutageSchedule</u>	A power system resource may have an outage schedule.
(0..n)	BusinessUnit	(0..1)	<u>BusinessUnit</u>	A power system resource may be operated by one BusinessUnit at a time.

A.2.2.12 Naming

This is a root class to provide common naming attributes for all classes needing naming attributes.

Naming Attributes

Native Attributes

aliasName	(String)	Free text name of the object or instance.
description	(String)	Description of the object or instance.
name	(String)	Unique name among objects owned by the same parent.
pathName	(String)	pathName is a concatenation of all names from each container.

A.2.2.13 PowerSystemResource

A power system resource can be an item of equipment such as a Switch, an EquipmentContainer containing many individual items of equipment such as a Substation, or an organisational entity such as Company or SubControlArea. This provides for the nesting of collections of PowerSystemResources within other PowerSystemResources. For example, a Switch could be a member of a Substation and a Substation could be a member of a division of a Company.

PowerSystemResource Attributes

Inherited Attributes

Naming.aliasName	(String)	Free text name of the object or instance.
Naming.description	(String)	Description of the object or instance.
Naming.name	(String)	Unique name among objects owned by the same parent.
Naming.pathName	(String)	pathName is a concatenation of all names from each container.

PowerSystemResource Associations

Native Roles

(0..n)	<i>OperatedBy_Companies</i>	(0..n)	<u>Company</u>	A power system resource may be part of one or more companies.
(0..n)	<i>PSRType</i>	(0..1)	<u>PSRType</u>	
(0..1)	<i>Contains_Measurements</i>	(0..n)	<u>Measurement</u>	Measurement-PSR defines the measurements in the naming hierarchy.
(1)	<i>OutageSchedule</i>	(0..1)	<u>OutageSchedule</u>	A power system resource may have an outage schedule.
(0..n)	<i>BusinessUnit</i>	(0..1)	<u>BusinessUnit</u>	A power system resource may be operated by one BusinessUnit at a time.

A.2.2.14 PSRType

Classifying instances of the same class, for example overhead and underground ACLineSegments. This classification mechanism is intended to provide flexibility outside the scope of this standard, i.e. provide customisation that is non standard.

PSRType Attributes

Inherited Attributes

Naming.aliasName	(String)	Free text name of the object or instance.
Naming.description	(String)	Description of the object or instance.
Naming.name	(String)	Unique name among objects owned by the same parent.
Naming.pathName	(String)	pathName is a concatenation of all names from each container.

PSRType Associations

<i>Native Roles</i>			
(0..1)	<i>PowerSystemResource</i>	(0..n)	<u>PowerSystemResource</u>

A.2.2.15 SubControlArea

An area defined for the purpose of tracking interchange with surrounding areas via tie points; may or may not serve as a control area.

SubControlArea Attributes

<i>Inherited Attributes</i>			
Naming.aliasName	(String)	Free text name of the object or instance.	
Naming.description	(String)	Description of the object or instance.	
Naming.name	(String)	Unique name among objects owned by the same parent.	
Naming.pathName	(String)	pathName is a concatenation of all names from each container.	

SubControlArea Associations

<i>Native Roles</i>			
(0..1)	<i>Contain_Substations</i>	(1..n)	<u>Substation</u> A SubControlArea may contain one or more Substations. The association is used in the naming hierarchy.
(0..1)	<i>GeneratingUnits</i>	(0..n)	<u>GeneratingUnit</u> A GeneratingUnit injects energy into a SubControlArea.
(1..1)	<i>Export_Energy Transactions</i>	(0..n)	<u>Energy Transaction</u> Energy is transferred between interchange areas.
(1..1)	<i>Import_Energy Transactions</i>	(0..n)	<u>Energy Transaction</u> Energy is transferred between interchange areas.
(0..n)	<i>HostControlArea</i>	(1..1)	<u>HostControlArea</u> The interchange area may operate as a control area.
(1..1)	<i>PartOf</i>	(0..n)	<u>ServicePoint</u> A transmission path's service point is part of an interchange area.
(1..1)	<i>SideA_TieLines</i>	(0..n)	<u>TieLine</u> The SubControlArea is on the A side of a collection of metered points which define the SubControl Area's boundary for a ControlArea Operator or CustomerConsumer.
(1..1)	<i>SideB_TieLines</i>	(0..n)	<u>TieLine</u> The SubControlArea is on the B side of a collection of metered points which define the SubControl Area's boundary for a ControlArea Operator or CustomerConsumer.

<i>Roles Inherited From PowerSystemResource</i>			
(0..n)	<i>OperatedBy_Companies</i>	(0..n)	<u>Company</u> A power system resource may be part of one or more companies.
(0..n)	<i>PSRType</i>	(0..1)	<u>PSRType</u>
(0..1)	<i>Contains_Measurements</i>	(0..n)	<u>Measurement</u> Measurement-PSR defines the measurements in the naming hierarchy.
(1)	<i>OutageSchedule</i>	(0..1)	<u>OutageSchedule</u> A power system resource may have an outage schedule.
(0..n)	<i>BusinessUnit</i>	(0..1)	<u>BusinessUnit</u> A power system resource may be operated by one BusinessUnit at a time.

A.2.2.16 Substation

A collection of equipment for purposes other than generation or utilization, through which electric energy in bulk is passed for the purposes of switching or modifying its characteristics.

Substation Attributes

Inherited Attributes

Naming.aliasName	(String)	Free text name of the object or instance.
Naming.description	(String)	Description of the object or instance.
Naming.name	(String)	Unique name among objects owned by the same parent.
Naming.pathName	(String)	pathName is a concatenation of all names from each container.

Substation Associations

Native Roles

(0..n)	<u>LoadArea</u>	(0..1)	<u>LoadArea</u>	The association is used in the naming hierarchy.
(1)	<u>Contains_Voltage Levels</u>	(0..n)	<u>VoltageLevel</u>	
(0..1)	<u>Contains_Bays</u>	(0..n)	<u>Bay</u>	
(0..1)	<u>Contains_Composite Switches</u>	(0..n)	<u>Composite Switch</u>	

Roles Inherited From EquipmentContainer

(1)	<u>ConnectivityNodes</u>	(0..n)	<u>Connectivity Node</u>
(0..1)	<u>Contains_Equipments</u>	(0..n)	<u>Equipment</u>

Roles Inherited From PowerSystemResource

(0..n)	<u>OperatedBy_Companies</u>	(0..n)	<u>Company</u>	A power system resource may be part of one or more companies.
(0..n)	<u>PSRType</u>	(0..1)	<u>PSRType</u>	Measurement-PSR defines the measurements in the naming hierarchy.
(0..1)	<u>Contains_Measurements</u>	(0..n)	<u>Measurement</u>	
(1)	<u>OutageSchedule</u>	(0..1)	<u>Outage Schedule</u>	A power system resource may have an outage schedule.
(0..n)	<u>BusinessUnit</u>	(0..1)	<u>BusinessUnit</u>	A power system resource may be operated by one BusinessUnit at a time.

A.2.2.17 Terminal

An electrical connection point to a piece of conducting equipment. Terminals are connected at physical connection points called "connectivity nodes".

Terminal Attributes

Inherited Attributes

Naming.aliasName	(String)	Free text name of the object or instance.
Naming.description	(String)	Description of the object or instance.
Naming.name	(String)	Unique name among objects owned by the same parent.
Naming.pathName	(String)	pathName is a concatenation of all names from each container.

Terminal Associations

<i>Native Roles</i>			
(0..n)	<i>ConductingEquipment</i>	(1)	<u>ConductingEquipment</u>
ConductingEquipment has 1 or 2 terminals that may be connected to other ConductingEquipment terminals via ConnectivityNodes.			
(0..1)	<i>Measurements</i>	(0..n)	<u>Measurement</u>
One or more measurements may be associated with a terminal in the network. Measurement-Terminal defines where the measurement is placed in the network topology. Some Measurements represent quantities related to a particular sensor position, for example a voltage transformer (PT) at a busbar or a current transformer (CT) at the bar between a breaker and an isolator. The sensing position is captured by the Measurement – Terminal association that makes it possible to place the sensing position at a well defined place. The place is defined by the connection of the Terminal to ConductingEquipment.			
(0..n)	<i>ConnectivityNode</i>	(0..1)	<u>ConnectivityNode</u>
Terminals interconnect with zero impedance at a node. Measurements on a node apply to all of its terminals.			

A.2.2.18 Unit

Quantity being measured. The Unit.name shall be unique among all specified quantities and describe the quantity. The Unit.aliasName is meant to be used for localization.

Unit Attributes

<i>Inherited Attributes</i>		
Naming.aliasName	(String)	Free text name of the object or instance.
Naming.description	(String)	Description of the object or instance.
Naming.name	(String)	Unique name among objects owned by the same parent.
Naming.pathName	(String)	pathName is a concatenation of all names from each container.

Unit Associations

<i>Native Roles</i>		
(1)	<i>Measurements</i>	(0..n) <u>Measurement</u>
(1)	<i>Controls</i>	(0..n) <u>Control</u>
(1)	<i>ProtectionEquipments</i>	(0..n) <u>ProtectionEquipment</u>

A.2.2.19 VoltageLevel

A collection of equipment at one common system voltage forming a switchgear. The equipment typically consist of breakers, busbars, instrumentation, control, regulation and protection devices as well as assemblies of all these.

VoltageLevel Attributes

Native Attributes

highVoltageLimit	(Voltage)	The bus bar's high voltage limit in kV.
lowVoltageLimit	(Voltage)	The bus bar's low voltage limit in kV.

Inherited Attributes

Naming.aliasName	(String)	Free text name of the object or instance.
Naming.description	(String)	Description of the object or instance.
Naming.name	(String)	Unique name among objects owned by the same parent.
Naming.pathName	(String)	pathName is a concatenation of all names from each container.

VoltageLevel Associations

Native Roles

(0..1)	Contains_Bays	(0..n)	<u>Bay</u>	The association is used in the naming hierarchy.
(0..n)	BaseVoltage	(1)	<u>BaseVoltage</u>	

Roles Inherited From EquipmentContainer

(1)	ConnectivityNodes	(0..n)	<u>ConnectivityNode</u>	
(0..1)	Contains_Equipments	(0..n)	<u>Equipment</u>	

Roles Inherited From PowerSystemResource

(0..n)	OperatedBy_Companies	(0..n)	<u>Company</u>	A power system resource may be part of one or more companies.
(0..n)	PSRType	(0..1)	<u>PSRType</u>	
(0..1)	Contains_Measurements	(0..n)	<u>Measurement</u>	Measurement-PSR defines the measurements in the naming hierarchy.
(1)	OutageSchedule	(0..1)	<u>OutageSchedule</u>	A power system resource may have an outage schedule.
(0..n)	BusinessUnit	(0..1)	<u>BusinessUnit</u>	A power system resource may be operated by one BusinessUnit at a time.

A.2.3 Domain

The domain package is a data dictionary of quantities and units that define datatypes for attributes (properties) that may be used by any class in any other package.

This package contains the definition of primitive datatypes, including units of measure and permissible values. Each datatype contains a value attribute and an optional unit of measure, which is specified as a static variable initialized to the textual description of the unit of measure. The value of the "units" string may be country or customer specific. Typical values are given. Permissible values for enumerations are listed in the documentation for the attribute using UML constraint syntax inside curly braces. Lengths of variable strings are listed in the descriptive text where required.

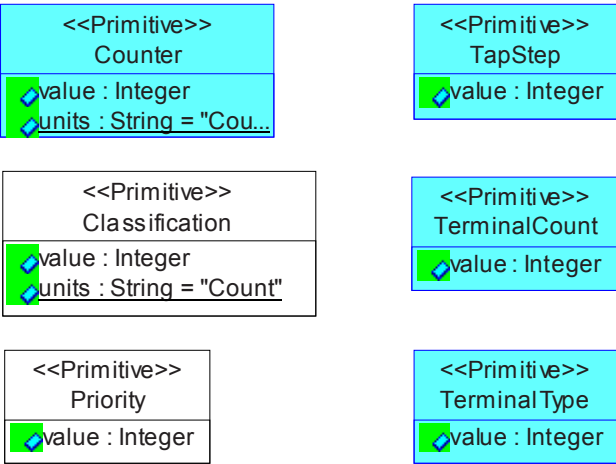


Figure A.4 – Integer datatypes

IEC 2623/03

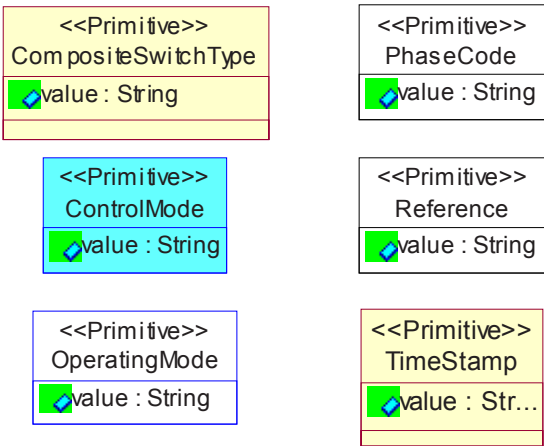
Figure A.4 shows all Integer datatypes.

<<Primitive>> ActivePower value : Float units : String = "MW"	<<Primitive>> EnergyAsMWh value : Float units : String = "MWh"		<<Primitive>> PUkPerMVA value : Float units : String = "PU_k/MVA..."
<<Primitive>> Admittance value : Float units : String = "Siemens"	<<Primitive>> ExcitingCurrent value : Float units : String = "PerCent"	<<Primitive>> Money value : Float units : String = "uoc"	<<Primitive>> RateOfChange value : Float units : String = "1 Hertz"
<<Primitive>> AngleDegrees value : Float units : String = "Degrees"	<<Primitive>> Exponent value : Float units : String = "Exponent"	<<Primitive>> NoLoadLoss value : Float units : String = "kW"	<<Primitive>> Ratio value : Float
<<Primitive>> AngleRadians value : Float units : String = "Radians"	<<Primitive>> Fraction value : Float	<<Primitive>> ParticipationFactor value : Float	<<Primitive>> Reactance value : Float units : String = "Ohms"
<<Primitive>> ApparentPower value : Float units : String = "MV..."	<<Primitive>> FreqBiasFactor value : Float units : String = "MW/0.1 H..."	<<Primitive>> PenaltyFactor value : Float	<<Primitive>> ReactivePower value : Float units : String = "MV..."
<<Primitive>> Conductance value : Float units : String = "Siemens"	<<Primitive>> Frequency value : Float units : String = "Hert..."	<<Primitive>> PerCent value : Float units : String = "PerCent"	<<Primitive>> Resistance value : Float units : String = "Ohms"
<<Primitive>> CostPerEnergyUnit value : Float units : String = "uoc/M..."	<<Primitive>> HeatPerHour value : Float units : String = "MBtu/Hou..."	<<Primitive>> PowerFactor value : Float units : String = "Ratio"	<<Primitive>> Seconds value : Float units : String = "Seconds"
<<Primitive>> CosPerHeatUnit value : Float units : String = "uoc/M..."	<<Primitive>> Hours value : Float units : String = "hours"	<<Primitive>> PowerROCPeMin value : Float units : String = "MW/minute"	<<Primitive>> ShortLength value : Float
<<Primitive>> CostPerHour value : Float units : String = "uoc/H..."	<<Primitive>> Impedance value : Float units : String = "ohms"	<<Primitive>> PowerROCPeSec value : Float units : String = "MW/second"	<<Primitive>> Susceptance value : Float units : String = "Siemens"
<<Primitive>> CurrentFlow value : Float units : String = "Amperes"	<<Primitive>> Inductance value : Float units : String = "Millihenries"	<<Primitive>> PowerVersusFrequency value : Float units : String = "PU_MW/PU_Frequenc..."	<<Primitive>> Temperature value : Float units : String = "TemperatureUnits"
<<Primitive>> Damping value : Float units : String = "PU_MW/PU_Frequenc..."	<<Primitive>> Inertia value : Float units : String = "PU_MW-Second"	<<Primitive>> PowerVersusVoltage value : Float units : String = "PU_MW/PU_kV"	<<Primitive>> Voltage value : Float units : String = "..."
<<Primitive>> Emission value : Float units : String = "kg/MBtu"	<<Primitive>> Load value : Float units : String = "MV..."	<<Primitive>> Pressure value : Float units : String = "lbf/in2"	<<Primitive>> Volume value : Float units : String = "Mm3"
	<<Primitive>> LoadLoss value : Float units : String = "kW"	<<Primitive>> PU value : Float	<<Primitive>> WaterLevel value : Float units : String = "m"
	<<Primitive>> LongLength value : Float units : String = "LongLength"		

IEC 2624/03

Figure A.5 – Float datatypes

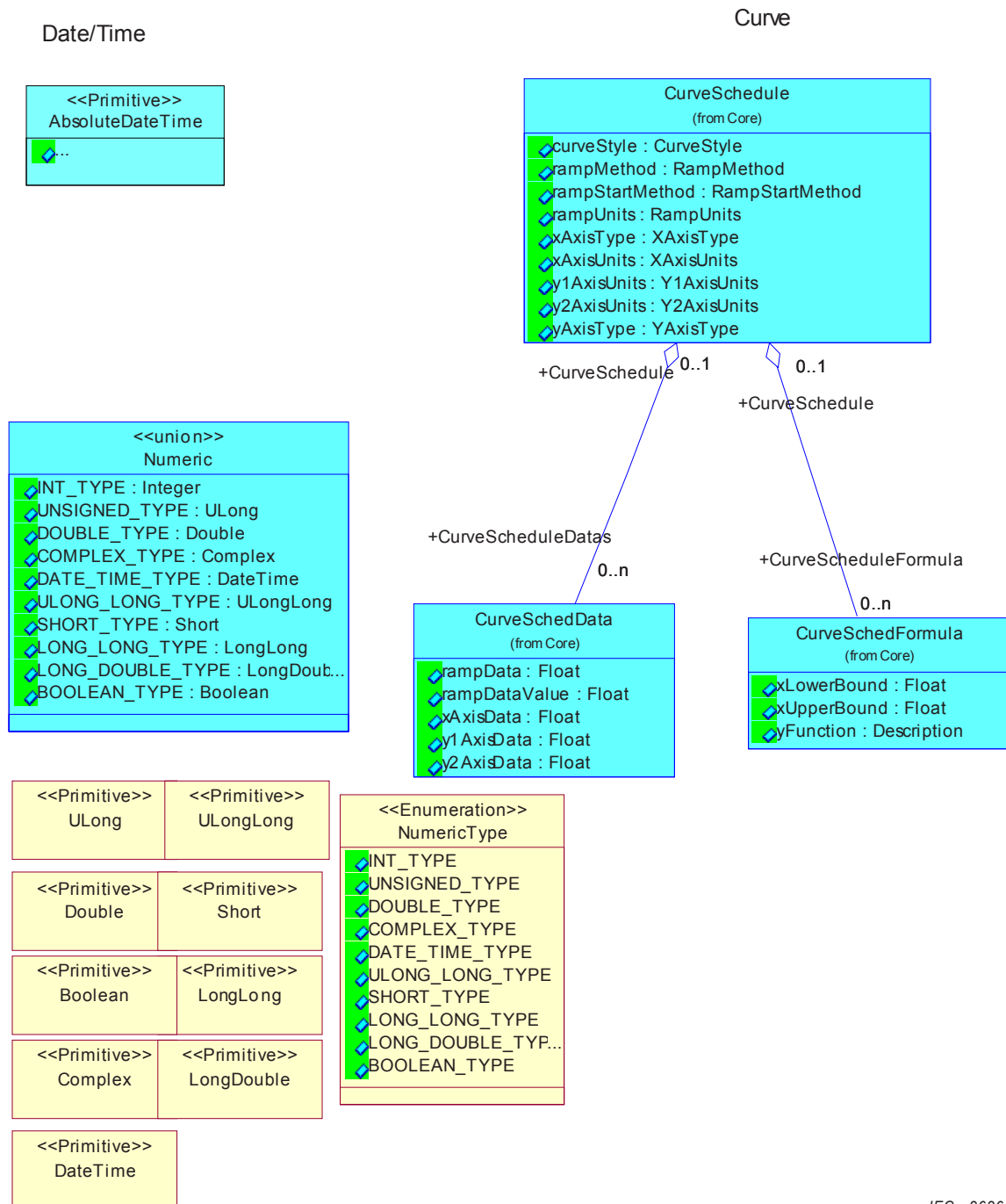
Figure A.5 shows all Float datatypes.



IEC 2625/03

Figure A.6 – String datatypes

Figure A.6 shows all String datatypes.



IEC 2626/03

Figure A.7 – Other datatypes

Figure A.7 shows all special datatypes.

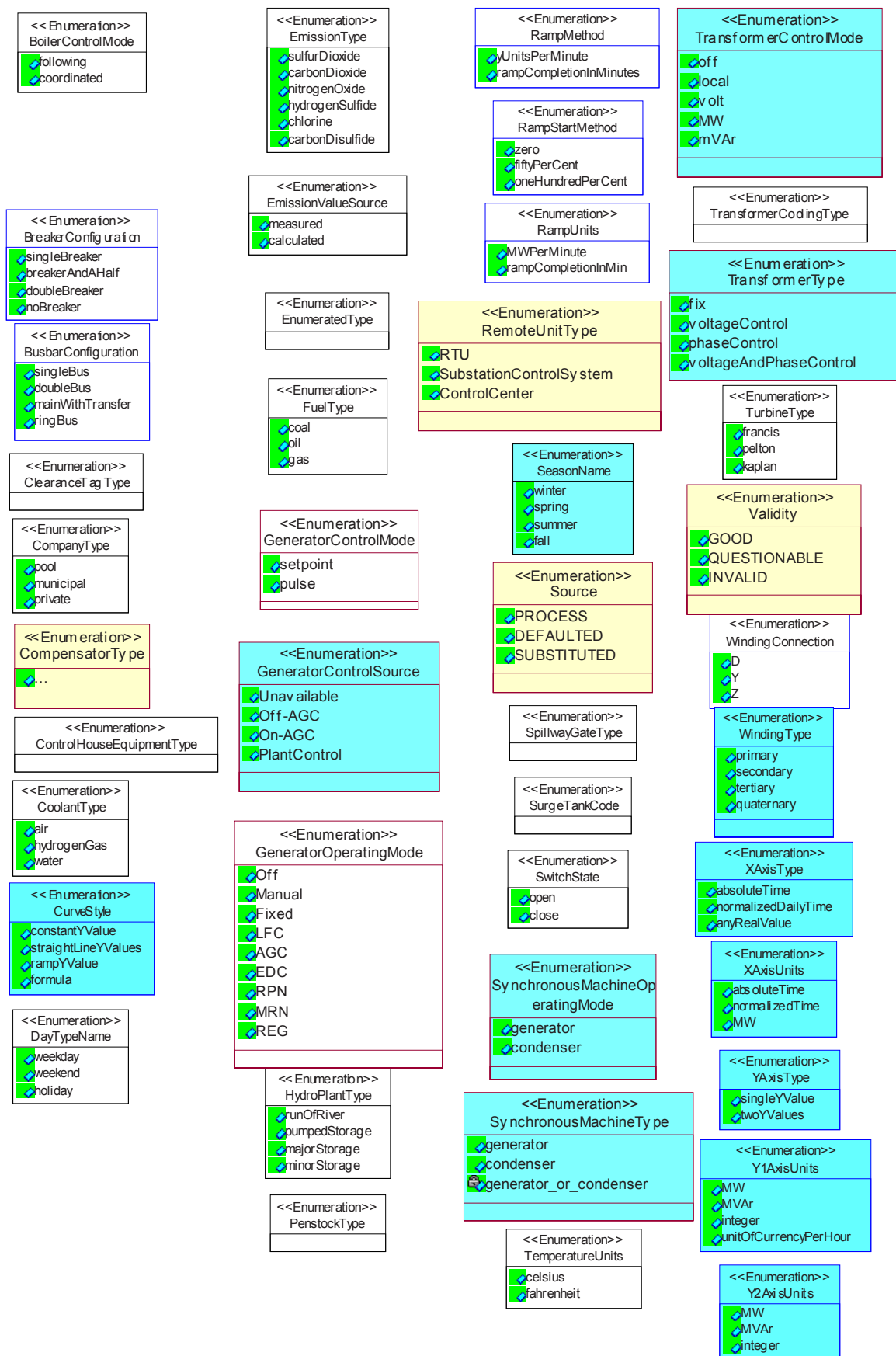


Figure A.8 – Enumeration datatypes

IEC 2627/03

Figure A.8 shows all Enumeration datatypes.

A.2.3.1 AbsoluteDateTime Primitive

Date and time as "yyyy-mm-ddThh:mm:ss", which conforms with ISO 8601 using the extended form and UTC rather than local time zones.

AbsoluteDateTime Attributes*Native Attributes*

value	(Timestamp)	Date and time as "yyyy-mm-ddThh:mm:ss", which conforms with ISO 8601 using the extended form and any of the zone options.
units	(String)	

A.2.3.2 ActivePower Primitive

Product of RMS value of the voltage and the RMS value of the in-phase component of the current (Megawatt).

ActivePower Attributes*Native Attributes*

value	(Float)
units	(String)

A.2.3.3 Admittance Primitive

Ratio of current to voltage.

Admittance Attributes*Native Attributes*

value	(Float)
units	(String)

A.2.3.4 AngleDegrees Primitive

Measurement of angle in degrees.

AngleDegrees Attributes*Native Attributes*

value	(Float)
units	(String)

A.2.3.5 AngleRadians Primitive

Phase angle in radians.

AngleRadians Attributes*Native Attributes*

value	(Float)
units	(String)

A.2.3.6 ApparentPower Primitive

Product of the RMS value of the voltage and the RMS value of the current (MegaVoltAmperes).

ApparentPower Attributes*Native Attributes*

value	(Float)
units	(String)

A.2.3.7 BoilerControlMode Enumeration

Boiler { Following, Coordinated}.

BoilerControlMode Attributes*Native Attributes*

following	()
coordinated	()

A.2.3.8 Boolean Primitive**Boolean Attributes****A.2.3.9 BreakerConfiguration Enumeration**

Switching arrangement for Bay.

BreakerConfiguration Attributes*Native Attributes*

singleBreaker	()
breakerAndAHalf	()
doubleBreaker	()
noBreaker	()

A.2.3.10 BusbarConfiguration Enumeration

Busbar layout for Bay.

BusbarConfiguration Attributes*Native Attributes*

singleBus	()
doubleBus	()
mainWithTransfer	()
ringBus	()

A.2.3.11 Classification Primitive

1..n, with 1 the most detailed, highest priority, etc.

Classification Attributes*Native Attributes*

value	(Integer)
units	(String)

A.2.3.12 ClearanceTagType Enumeration

Type of ClearanceTag. Could indicate the type of work to be performed and/or the type of supervisory control.

ClearanceTagType Attributes

A.2.3.13 CompanyType Enumeration

Type of company.

CompanyType Attributes

Native Attributes

pool	()
municipal	()
private	()

A.2.3.14 CompensatorType Enumeration

CompensatorType Attributes

Native Attributes

shunt	()
series	()

A.2.3.15 Complex Primitive

Complex Attributes

A.2.3.16 CompositeSwitchType Primitive

An alphanumeric code that can be used as a reference to extra information such as the description of the interlocking scheme if any.

CompositeSwitchType Attributes

Native Attributes

value	(String)
-------	----------

A.2.3.17 Conductance Primitive

Factor by which voltage must be multiplied to give corresponding power lost from a circuit. Real part of admittance.

Conductance Attributes

Native Attributes

value	(Float)
units	(String)

A.2.3.18 ControlHouseEquipmentType Enumeration

Type of control house equipment.

ControlHouseEquipmentType Attributes

A.2.3.19 ControlMode Primitive

Textual name for a control mode.

ControlMode Attributes

Native Attributes

value	(String)
-------	----------

A.2.3.20 CoolantType Enumeration

Method of cooling a machine.

CoolantType Attributes

Native Attributes

air	()
hydrogenGas	()
water	()

A.2.3.21 CostPerEnergyUnit Primitive

Cost, in units of currency, per megawatt hour of electricity generated.

CostPerEnergyUnit Attributes

Native Attributes

value	(Float)
units	(String)

A.2.3.22 CostPerHeatUnit Primitive

Cost, in units of currency, per million Btus of heat generated.

CostPerHeatUnit Attributes

Native Attributes

value	(Float)
units	(String)

A.2.3.23 CostPerHour Primitive

Cost, in units of currency, per hour of elapsed time.

CostPerHour Attributes

Native Attributes

value	(Float)
units	(String)

A.2.3.24 Counter Primitive

Measurement of quantity (must be greater than or equal to zero).

Counter Attributes

Native Attributes

value	(Integer)	Value must be greater than or equal to 0.
units	(String)	

A.2.3.25 CurrentFlow Primitive

Current flow in Amps (positive flow is out of the ConductingEquipment into the ConnectivityNode).

CurrentFlow Attributes

Native Attributes

value	(Float)
units	(String)

A.2.3.26 CurveStyle Enumeration

Style or shape of curve.

CurveStyle Attributes

Native Attributes

constantYValue	()
straightLineYValues	()
rampYValue	()
formula	()

A.2.3.27 Damping Primitive

Per-unit megawatt variation with per-unit frequency referenced on the system MVA base.
Typical values in range 1,0 to 2,0

Damping Attributes

Native Attributes

value	(Float)
units	(String)

A.2.3.28 DateTime Primitive

DateTime Attributes

A.2.3.29 DayTypeName Enumeration

Name of day type.

DayTypeName Attributes

Native Attributes

weekday	()
weekend	()
holiday	()

A.2.3.30 Double Primitive

Double Attributes

A.2.3.31 Emission Primitive

Quantity of emission per fuel heat content.

Emission Attributes

Native Attributes

value	(Float)
units	(String)

A.2.3.32 EmissionType Enumeration

The type of emission.

EmissionType Attributes

Native Attributes

sulphurDioxide	()
carbonDioxide	()
nitrogenOxide	()
hydrogenSulfide	()

chlorine	()
carbonDisulfide	()

A.2.3.33 EmissionValueSource Enumeration

The source of the emission value.

EmissionValueSource Attributes

Native Attributes

measured	()
calculated	()

A.2.3.34 EnergyAsMWh Primitive

Electric energy, in megawatt-hours.

EnergyAsMWh Attributes

Native Attributes

value	(Float)
units	(String)

A.2.3.35 EnumeratedType Enumeration

An enumerated type for attributes whose domain is a finite set of values. The legal values are normally listed in the attribute description, using the UML constraint syntax {value1, value2, ...valueN}.

EnumeratedType Attributes

A.2.3.36 ExcitingCurrent Primitive

The exciting current on open-circuit test, expressed as a percentage of rated current, at nominal voltage.

ExcitingCurrent Attributes

Native Attributes

value	(Float)
units	(String)

A.2.3.37 Exponent Primitive

Exponentiation amount.

Exponent Attributes

Native Attributes

value	(Float)
units	(String)

A.2.3.38 Fraction Primitive

Fractional amount. Sum of all fractions of the same quantity should be 1,0.

Fraction Attributes

Native Attributes

value	(Float)
-------	---------