IEC 61968-11

Edition 2.0 2013-03

INTERNATIONAL STANDARD

NORME INTERNATIONALE

colour inside

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

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

IEC 61968-11:2013

About the IEC

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

About IEC publications

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

Useful links:

IEC publications search - www.iec.ch/searchpub

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It also gives information on projects, replaced and withdrawn publications.

IEC Just Published - webstore.iec.ch/justpublished

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

Electropedia - www.electropedia.org

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

Customer Service Centre - webstore.iec.ch/csc

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

A propos de la CEI

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

A propos des publications CEI

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

Liens utiles:

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

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

Just Published CEI - webstore.iec.ch/justpublished

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

Electropedia - www.electropedia.org

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

Service Clients - webstore.iec.ch/csc

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

IEC 61968-11

Edition 2.0 2013-03

INTERNATIONAL STANDARD

NORME INTERNATIONALE

colour inside

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

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

INTERNATIONAL
ELECTROTECHNICAL
COMMISSION

COMMISSION
ELECTROTECHNIQUE
INTERNATIONALE

PRICE CODE
CODE PRIX XH

ICS 33.200 ISBN 978-2-83220-662-1

Warning! Make sure that you obtained this publication from an authorized distributor.
Attention! Veuillez vous assurer que vous avez obtenu cette publication via un distributeur agréé.

CONTENTS

FOREWORD ... 15
INTRODUCTION ... 17
1 Scope ... 18
2 Normative references ... 18
3 Terms and definitions ... 19
4 CIM specification ... 20
4.1 CIM modelling notation ... 20
4.2 CIM packages ... 21
4.2.1 General ... 21
4.2.2 CIM packages ... 21
4.2.3 CIM extensions for distribution packages (this part of IEC 61968) ... 22
4.3 CIM UML modelling ... 23
4.3.1 General ... 23
4.3.2 Scope of UML model ... 23
4.3.3 Extensibility ... 23
4.3.4 Message payload (profile) definition ... 24
4.4 DCIM model concepts and examples ... 25
4.4.1 General ... 25
4.4.2 Common concepts ... 25
4.4.3 Network modelling concepts and examples ... 32
4.4.4 Customers model ... 53
4.4.5 Metering model ... 54
4.4.6 Payment metering ... 64
4.5 Other ... 69
5 Detailed model ... 69
5.1 Overview ... 69
5.2 Context ... 69
6 Top package IEC61968 ... 70
6.1 General ... 70
6.2 IEC61968CIMVersion root class ... 71
6.3 Package Common ... 72
6.3.1 General ... 72
6.3.2 Status compound ... 73
6.3.3 PostalAddress compound ... 74
6.3.4 StreetAddress compound ... 74
6.3.5 StreetDetail compound ... 74
6.3.6 TownDetail compound ... 75
6.3.7 ElectronicAddress compound ... 75
6.3.8 TelephoneNumber compound ... 76
6.3.9 ActivityRecord ... 76
6.3.10 Agreement ... 77
6.3.11 ConfigurationEvent ... 77
6.3.12 CoordinateSystem ... 78
6.3.13 Document ... 79
6.3.14 Location ... 80

6.3.15 Organisation ... 81
6.3.16 OrganisationRole ... 81
6.3.17 PositionPoint root class ... 82
6.3.18 TimePoint ... 83
6.3.19 TimeSchedule ... 83
6.3.20 UserAttribute root class ... 84
6.4 Package Assets ... 85
6.4.1 General ... 85
6.4.2 AcceptanceTest compound ... 86
6.4.3 LifecycleDate compound ... 87
6.4.4 AssetModelUsageKind enumeration ... 87
6.4.5 CorporateStandardKind enumeration ... 87
6.4.6 SealConditionKind enumeration ... 88
6.4.7 SealKind enumeration ... 88
6.4.8 Asset ... 88
6.4.9 AssetContainer ... 89
6.4.10 AssetFunction ... 91
6.4.11 AssetInfo ... 91
6.4.12 AssetModel ... 92
6.4.13 AssetOrganisationRole ... 92
6.4.14 AssetOwner ... 93
6.4.15 ComMedia ... 93
6.4.16 Manufacturer ... 94
6.4.17 ProductAssetModel ... 95
6.4.18 Seal ... 96
6.5 Package AssetInfo ... 96
6.5.1 General ... 96
6.5.2 BusbarSectionInfo ... 100
6.5.3 CableConstructionKind enumeration ... 101
6.5.4 CableInfo ... 101
6.5.5 CableOuterJacketKind enumeration ... 102
6.5.6 CableShieldMaterialKind enumeration ... 103
6.5.7 ConcentricNeutralCableInfo ... 103
6.5.8 NoLoadTest ... 104
6.5.9 OpenCircuitTest ... 105
6.5.10 OverheadWireInfo ... 106
6.5.11 PowerTransformerInfo ... 107
6.5.12 ShortCircuitTest ... 107
6.5.13 SwitchInfo ... 108
6.5.14 TapChangerInfo ... 109
6.5.15 TapeShieldCableInfo ... 110
6.5.16 TransformerEndInfo ... 111
6.5.17 TransformerTankInfo ... 112
6.5.18 TransformerTest ... 113
6.5.19 WireInfo ... 113
6.5.20 WireInsulationKind enumeration ... 115
6.5.21 WireMaterialKind enumeration ... 115
6.5.22 WirePosition ... 116
6.5.23 WireSpacingInfo ... 116

6.5.24 WireUsageKind enumeration ... 117
6.6 Package Work ... 117
6.6.1 General ... 117
6.6.2 WorkKind enumeration ... 118
6.6.3 Work ... 119
6.7 Package Customers ... 119
6.7.1 General ... 119
6.7.2 CustomerKind enumeration ... 121
6.7.3 RevenueKind enumeration ... 121
6.7.4 ServiceKind enumeration ... 121
6.7.5 Customer ... 122
6.7.6 CustomerAccount ... 123
6.7.7 CustomerAgreement ... 124
6.7.8 PricingStructure ... 125
6.7.9 ServiceCategory ... 126
6.7.10 ServiceLocation ... 127
6.7.11 Tariff ... 128
6.8 Package Metering ... 128
6.8.1 General ... 128
6.8.2 AmiBillingReadyKind enumeration ... 138
6.8.3 ComDirectionKind enumeration ... 139
6.8.4 ComTechnologyKind enumeration ... 139
6.8.5 EndDeviceFunctionKind enumeration ... 140
6.8.6 MeterMultiplierKind enumeration ... 140
6.8.7 RandomisationKind enumeration ... 140
6.8.8 ReadingReasonKind enumeration ... 141
6.8.9 ServiceMultiplierKind enumeration ... 141
6.8.10 TransmissionModeKind enumeration ... 142
6.8.11 UsagePointConnectedKind enumeration ... 142
6.8.12 ControlledAppliance compound ... 142
6.8.13 EndDeviceCapability compound ... 143
6.8.14 EndDeviceTiming compound ... 144
6.8.15 RationalNumber compound ... 144
6.8.16 ReadingInterharmonic compound ... 144
6.8.17 BaseReading ... 144
6.8.18 Channel ... 145
6.8.19 ComFunction ... 146
6.8.20 ComModule ... 147
6.8.21 DemandResponseProgram ... 148
6.8.22 EndDevice ... 148
6.8.23 EndDeviceAction root class ... 150
6.8.24 EndDeviceControl ... 150
6.8.25 EndDeviceControlType ... 152
6.8.26 EndDeviceEvent ... 153
6.8.27 EndDeviceEventDetail root class ... 153
6.8.28 EndDeviceEventType ... 154
6.8.29 EndDeviceFunction ... 155
6.8.30 EndDeviceGroup ... 155
6.8.31 EndDeviceInfo ... 156

6.8.32 IntervalBlock root class ... 156
6.8.33 IntervalReading ... 157
6.8.34 Meter ... 158
6.8.35 MeterMultiplier ... 159
6.8.36 MeterReading ... 160
6.8.37 MeterServiceWork ... 160
6.8.38 MetrologyRequirement ... 161
6.8.39 PanDemandResponse ... 162
6.8.40 PanDisplay ... 163
6.8.41 PanPricing ... 164
6.8.42 PanPricingDetail root class ... 164
6.8.43 PendingCalculation root class ... 165
6.8.44 Reading ... 166
6.8.45 ReadingQuality root class ... 167
6.8.46 ReadingQualityType ... 167
6.8.47 ReadingType ... 168
6.8.48 Register ... 170
6.8.49 ServiceMultiplier ... 170
6.8.50 SimpleEndDeviceFunction ... 171
6.8.51 UsagePoint ... 171
6.8.52 UsagePointGroup ... 173
6.8.53 UsagePointLocation ... 174
6.9 Package LoadControl ... 175
6.9.1 General ... 175
6.9.2 RemoteConnectDisconnectInfo compound ... 176
6.9.3 ConnectDisconnectFunction ... 176
6.10 Package PaymentMetering ... 178
6.10.1 General ... 178
6.10.2 AccountMovement compound ... 184
6.10.3 AccountingUnit compound ... 185
6.10.4 BankAccountDetail compound ... 185
6.10.5 Due compound ... 185
6.10.6 LineDetail compound ... 186
6.10.7 ChargeKind enumeration ... 186
6.10.8 ChequeKind enumeration ... 186
6.10.9 SupplierKind enumeration ... 187
6.10.10 TenderKind enumeration ... 187
6.10.11 TransactionKind enumeration ... 187
6.10.12 AuxiliaryAccount ... 188
6.10.13 AuxiliaryAgreement ... 188
6.10.14 Card root class ... 190
6.10.15 Cashier ... 190
6.10.16 CashierShift ... 191
6.10.17 Charge ... 192
6.10.18 Cheque root class ... 193
6.10.19 ConsumptionTariffInterval root class ... 193
6.10.20 MerchantAccount ... 194
6.10.21 MerchantAgreement ... 195
6.10.22 PointOfSale ... 195

6.10.23 Receipt 196
6.10.24 ServiceSupplier 197
6.10.25 Shift 197
6.10.26 TariffProfile 198
6.10.27 Tender 199
6.10.28 TimeTariffInterval root class 200
6.10.29 Transaction 201
6.10.30 Transactor 202
6.10.31 Vendor 203
6.10.32 VendorShift 203
Bibliography 205

Figure 1 – CIM packages 21
Figure 2 – CIM extensions for distribution (DCIM) top-level packages 22
Figure 3 – DCIM key classes 25
Figure 4 – DCIM power system resource and asset locations 27
Figure 5 – DCIM organization model 28
Figure 6 – DCIM assets model 29
Figure 7 – DCIM assets – specialising guidelines 31
Figure 8 – DCIM phase modelling 33
Figure 9 – DCIM load model 35
Figure 10 – DCIM line connectivity model 37
Figure 11 – DCIM conductor (line and cable datasheet) model 39
Figure 12 – DCIM transformer connectivity model 43
Figure 13 – DCIM transformer datasheet model 46
Figure 14 – DCIM tap changer model 48
Figure 15 – Example of a distribution transformer that can be modelled with DCIM 49
Figure 16 – Example of a two-winding transformer connected as an autotransformer 50
Figure 17 – DCIM auxiliary equipment 52
Figure 18 – DCIM customers model 53
Figure 19 – DCIM metering model 55
Figure 20 – DCIM usage point model 56
Figure 21 – DCIM end device model 58
Figure 22 – Configuration events for metering 60
Figure 23 – DCIM meter readings model 61
Figure 24 – DCIM end device controls and events model 63
Figure 25 – DCIM transacting model 65
Figure 26 – DCIM receipting model 66
Figure 27 – DCIM auxiliary agreement model 67
Figure 28 – DCIM pricing structure model 68
Figure 29 – Package diagram IEC61968::IEC61968Dependencies 71
Figure 30 – Class diagram Common::CommonInheritance 72
Figure 31 – Class diagram Common::CommonOverview 73
Figure 32 – Class diagram Assets::AssetsInheritance 85

Figure 33 – Class diagram Assets::AssetsOverview ... 86
Figure 34 – Class diagram AssetInfo::AssetInfoInheritance ... 97
Figure 35 – Class diagram AssetInfo::AssetInfoOverview ... 98
Figure 36 – Class diagram AssetInfo::DCIMWireInfo ... 99
Figure 37 – Class diagram AssetInfo::DCIMTransformerInfo ... 100
Figure 38 – Class diagram Work::WorkInheritance ... 118
Figure 39 – Class diagram Work::WorkOverview ... 118
Figure 40 – Class diagram Customers::CustomersInheritance ... 120
Figure 41 – Class diagram Customers::CustomersOverview ... 120
Figure 42 – Class diagram Metering::MeteringInheritance ... 129
Figure 43 – Class diagram Metering::MeteringDatatypes ... 130
Figure 44 – Class diagram Metering::MeteringOverviewShort ... 131
Figure 45 – Class diagram Metering::MeteringUsagePoints ... 132
Figure 46 – Class diagram Metering::MeteringEndDevices ... 133
Figure 47 – Class diagram Metering::MeteringConfigurationEvents ... 134
Figure 48 – Class diagram Metering::MeteringMeterReadings ... 135
Figure 49 – Class diagram Metering::MeteringEventsAndControls ... 136
Figure 50 – Class diagram Metering::MeteringMultipliers ... 137
Figure 51 – Class diagram Metering::MeteringTypes ... 138
Figure 52 – Class diagram LoadControl::LoadControlInheritance ... 175
Figure 53 – Class diagram LoadControl::LoadControlOverview ... 176
Figure 54 – Class diagram PaymentMetering::PaymentMeteringInheritance ... 178
Figure 55 – Class diagram PaymentMetering::PaymentMeteringOverview ... 179
Figure 56 – Class diagram PaymentMetering::PaymentMeteringRelationships ... 180
Figure 57 – Class diagram PaymentMetering::Transacting ... 181
Figure 58 – Class diagram PaymentMetering::Receipting ... 182
Figure 59 – Class diagram PaymentMetering::AuxiliaryAgreement ... 183
Figure 60 – Class diagram PaymentMetering::TariffProfile ... 184

Table 1 – Open wye/open delta transformer bank connections ... 49
Table 2 – Attribute documentation ... 69
Table 3 – Association ends documentation ... 70
Table 4 – Enums documentation ... 70
Table 5 – Attributes of IEC61968::IEC61968CIMVersion ... 71
Table 6 – Attributes of Common::Status ... 74
Table 7 – Attributes of Common::PostalAddress ... 74
Table 8 – Attributes of Common::StreetAddress ... 74
Table 9 – Attributes of Common::StreetDetail ... 74
Table 10 – Attributes of Common::TownDetail ... 75
Table 11 – Attributes of Common::ElectronicAddress ... 75
Table 12 – Attributes of Common::TelephoneNumber ... 76
Table 13 – Attributes of Common::ActivityRecord ... 76
Table 14 – Association ends of Common::ActivityRecord with other classes ... 76

Table 15 – Attributes of Common::Agreement 77
Table 16 – Association ends of Common::Agreement with other classes 77
Table 17 – Attributes of Common::ConfigurationEvent 77
Table 18 – Association ends of Common::ConfigurationEvent with other classes 78
Table 19 – Attributes of Common::CoordinateSystem 78
Table 20 – Association ends of Common::CoordinateSystem with other classes 79
Table 21 – Attributes of Common::Document 79
Table 22 – Association ends of Common::Document with other classes 80
Table 23 – Attributes of Common::Location 80
Table 24 – Association ends of Common::Location with other classes 80
Table 25 – Attributes of Common::Organisation 81
Table 26 – Association ends of Common::Organisation with other classes 81
Table 27 – Attributes of Common::OrganisationRole 82
Table 28 – Association ends of Common::OrganisationRole with other classes 82
Table 29 – Attributes of Common::PositionPoint 82
Table 30 – Association ends of Common::PositionPoint with other classes 82
Table 31 – Attributes of Common::TimePoint 83
Table 32 – Association ends of Common::TimePoint with other classes 83
Table 33 – Attributes of Common::TimeSchedule 83
Table 34 – Association ends of Common::TimeSchedule with other classes 84
Table 35 – Attributes of Common::UserAttribute 84
Table 36 – Association ends of Common::UserAttribute with other classes 85
Table 37 – Attributes of Assets::AcceptanceTest 86
Table 38 – Attributes of Assets::LifecycleDate 87
Table 39 – Literals of Assets::AssetModelUsageKind 87
Table 40 – Literals of Assets::CorporateStandardKind 88
Table 41 – Literals of Assets::SealConditionKind 88
Table 42 – Literals of Assets::SealKind 88
Table 43 – Attributes of Assets::Asset 89
Table 44 – Association ends of Assets::Asset with other classes 89
Table 45 – Attributes of Assets::AssetContainer 90
Table 46 – Association ends of Assets::AssetContainer with other classes 90
Table 47 – Attributes of Assets::AssetFunction 91
Table 48 – Association ends of Assets::AssetFunction with other classes 91
Table 49 – Attributes of Assets::AssetInfo 91
Table 50 – Association ends of Assets::AssetInfo with other classes 92
Table 51 – Attributes of Assets::AssetModel 92
Table 52 – Association ends of Assets::AssetModel with other classes 92
Table 53 – Attributes of Assets::AssetOrganisationRole 92
Table 54 – Association ends of Assets::AssetOrganisationRole with other classes 93
Table 55 – Attributes of Assets::AssetOwner 93
Table 56 – Association ends of Assets::AssetOwner with other classes 93
Table 57 – Attributes of Assets::ComMedia 93

Table 58 – Association ends of Assets::ComMedia with other classes 94
Table 59 – Attributes of Assets::Manufacturer 94
Table 60 – Association ends of Assets::Manufacturer with other classes 95
Table 61 – Attributes of Assets::ProductAssetModel 95
Table 62 – Association ends of Assets::ProductAssetModel with other classes 95
Table 63 – Attributes of Assets::Seal 96
Table 64 – Association ends of Assets::Seal with other classes 96
Table 65 – Attributes of AssetInfo::BusbarSectionInfo 100
Table 66 – Association ends of AssetInfo::BusbarSectionInfo with other classes 101
Table 67 – Literals of AssetInfo::CableConstructionKind 101
Table 68 – Attributes of AssetInfo::CableInfo 101
Table 69 – Association ends of AssetInfo::CableInfo with other classes 102
Table 70 – Literals of AssetInfo::CableOuterJacketKind 102
Table 71 – Literals of AssetInfo::CableShieldMaterialKind 103
Table 72 – Attributes of AssetInfo::ConcentricNeutralCableInfo 103
Table 73 – Association ends of AssetInfo::ConcentricNeutralCableInfo with other classes 104
Table 74 – Attributes of AssetInfo::NoLoadTest 104
Table 75 – Association ends of AssetInfo::NoLoadTest with other classes 105
Table 76 – Attributes of AssetInfo::OpenCircuitTest 105
Table 77 – Association ends of AssetInfo::OpenCircuitTest with other classes 106
Table 78 – Attributes of AssetInfo::OverheadWireInfo 106
Table 79 – Association ends of AssetInfo::OverheadWireInfo with other classes 107
Table 80 – Attributes of AssetInfo::PowerTransformerInfo 107
Table 81 – Association ends of AssetInfo::PowerTransformerInfo with other classes 107
Table 82 – Attributes of AssetInfo::ShortCircuitTest 108
Table 83 – Association ends of AssetInfo::ShortCircuitTest with other classes 108
Table 84 – Attributes of AssetInfo::SwitchInfo 108
Table 85 – Association ends of AssetInfo::SwitchInfo with other classes 109
Table 86 – Attributes of AssetInfo::TapChangerInfo 109
Table 87 – Association ends of AssetInfo::TapChangerInfo with other classes 109
Table 88 – Attributes of AssetInfo::TapeShieldCableInfo 110
Table 89 – Association ends of AssetInfo::TapeShieldCableInfo with other classes 111
Table 90 – Attributes of AssetInfo::TransformerEndInfo 111
Table 91 – Association ends of AssetInfo::TransformerEndInfo with other classes 112
Table 92 – Attributes of AssetInfo::TransformerTankInfo 112
Table 93 – Association ends of AssetInfo::TransformerTankInfo with other classes 113
Table 94 – Attributes of AssetInfo::TransformerTest 113
Table 95 – Association ends of AssetInfo::TransformerTest with other classes 113
Table 96 – Attributes of AssetInfo::WireInfo 114
Table 97 – Association ends of AssetInfo::WireInfo with other classes 114
Table 98 – Literals of AssetInfo::WireInsulationKind 115
Table 99 – Literals of AssetInfo::WireMaterialKind 115

Table 100 – Attributes of AssetInfo::WirePosition 116
Table 101 – Association ends of AssetInfo::WirePosition with other classes 116
Table 102 – Attributes of AssetInfo::WireSpacingInfo 116
Table 103 – Association ends of AssetInfo::WireSpacingInfo with other classes 117
Table 104 – Literals of AssetInfo::WireUsageKind 117
Table 105 – Literals of Work::WorkKind 118
Table 106 – Attributes of Work::Work 119
Table 107 – Association ends of Work::Work with other classes 119
Table 108 – Literals of Customers::CustomerKind 121
Table 109 – Literals of Customers::RevenueKind 121
Table 110 – Literals of Customers::ServiceKind 122
Table 111 – Attributes of Customers::Customer 122
Table 112 – Association ends of Customers::Customer with other classes 122
Table 113 – Attributes of Customers::CustomerAccount 123
Table 114 – Association ends of Customers::CustomerAccount with other classes 123
Table 115 – Attributes of Customers::CustomerAgreement 124
Table 116 – Association ends of Customers::CustomerAgreement with other classes 124
Table 117 – Attributes of Customers::PricingStructure 125
Table 118 – Association ends of Customers::PricingStructure with other classes 126
Table 119 – Attributes of Customers::ServiceCategory 126
Table 120 – Association ends of Customers::ServiceCategory with other classes 126
Table 121 – Attributes of Customers::ServiceLocation 127
Table 122 – Association ends of Customers::ServiceLocation with other classes 127
Table 123 – Attributes of Customers::Tariff 128
Table 124 – Association ends of Customers::Tariff with other classes 128
Table 125 – Literals of Metering::AmiBillingReadyKind 139
Table 126 – Literals of Metering::ComDirectionKind 139
Table 127 – Literals of Metering::ComTechnologyKind 139
Table 128 – Literals of Metering::EndDeviceFunctionKind 140
Table 129 – Literals of Metering::MeterMultiplierKind 140
Table 130 – Literals of Metering::RandomisationKind 141
Table 131 – Literals of Metering::ReadingReasonKind 141
Table 132 – Literals of Metering::ServiceMultiplierKind 142
Table 133 – Literals of Metering::TransmissionModeKind 142
Table 134 – Literals of Metering::UsagePointConnectedKind 142
Table 135 – Attributes of Metering::ControlledAppliance 143
Table 136 – Attributes of Metering::EndDeviceCapability 143
Table 137 – Attributes of Metering::EndDeviceTiming 144
Table 138 – Attributes of Metering::RationalNumber 144
Table 139 – Attributes of Metering::ReadingInterharmonic 144
Table 140 – Attributes of Metering::BaseReading 145
Table 141 – Association ends of Metering::BaseReading with other classes 145
Table 142 – Attributes of Metering::Channel 145

Table 143 – Association ends of Metering::Channel with other classes 146
Table 144 – Attributes of Metering::ComFunction 146
Table 145 – Association ends of Metering::ComFunction with other classes 146
Table 146 – Attributes of Metering::ComModule 147
Table 147 – Association ends of Metering::ComModule with other classes 147
Table 148 – Attributes of Metering::DemandResponseProgram 148
Table 149 – Association ends of Metering::DemandResponseProgram with other classes 148
Table 150 – Attributes of Metering::EndDevice 149
Table 151 – Association ends of Metering::EndDevice with other classes 149
Table 152 – Attributes of Metering::EndDeviceAction 150
Table 153 – Association ends of Metering::EndDeviceAction with other classes 150
Table 154 – Attributes of Metering::EndDeviceControl 151
Table 155 – Association ends of Metering::EndDeviceControl with other classes 152
Table 156 – Attributes of Metering::EndDeviceControlType 152
Table 157 – Association ends of Metering::EndDeviceControlType with other classes 152
Table 158 – Attributes of Metering::EndDeviceEvent 153
Table 159 – Association ends of Metering::EndDeviceEvent with other classes 153
Table 160 – Attributes of Metering::EndDeviceEventDetail 154
Table 161 – Association ends of Metering::EndDeviceEventDetail with other classes 154
Table 162 – Attributes of Metering::EndDeviceEventType 154
Table 163 – Association ends of Metering::EndDeviceEventType with other classes 154
Table 164 – Attributes of Metering::EndDeviceFunction 155
Table 165 – Association ends of Metering::EndDeviceFunction with other classes 155
Table 166 – Attributes of Metering::EndDeviceGroup 155
Table 167 – Association ends of Metering::EndDeviceGroup with other classes 156
Table 168 – Attributes of Metering::EndDeviceInfo 156
Table 169 – Association ends of Metering::EndDeviceInfo with other classes 156
Table 170 – Association ends of Metering::IntervalBlock with other classes 157
Table 171 – Attributes of Metering::IntervalReading 157
Table 172 – Association ends of Metering::IntervalReading with other classes 157
Table 173 – Attributes of Metering::Meter 158
Table 174 – Association ends of Metering::Meter with other classes 159
Table 175 – Attributes of Metering::MeterMultiplier 159
Table 176 – Association ends of Metering::MeterMultiplier with other classes 160
Table 177 – Attributes of Metering::MeterReading 160
Table 178 – Association ends of Metering::MeterReading with other classes 160
Table 179 – Attributes of Metering::MeterServiceWork 161
Table 180 – Association ends of Metering::MeterServiceWork with other classes 161
Table 181 – Attributes of Metering::MetrologyRequirement 161
Table 182 – Association ends of Metering::MetrologyRequirement with other classes 162
Table 183 – Attributes of Metering::PanDemandResponse 162
Table 184 – Association ends of Metering::PanDemandResponse with other classes 163

Table 185 – Attributes of Metering::PanDisplay 163
Table 186 – Association ends of Metering::PanDisplay with other classes 164
Table 187 – Attributes of Metering::PanPricing 164
Table 188 – Association ends of Metering::PanPricing with other classes 164
Table 189 – Attributes of Metering::PanPricingDetail 164
Table 190 – Association ends of Metering::PanPricingDetail with other classes 165
Table 191 – Attributes of Metering::PendingCalculation 165
Table 192 – Association ends of Metering::PendingCalculation with other classes 166
Table 193 – Attributes of Metering::Reading 166
Table 194 – Association ends of Metering::Reading with other classes 166
Table 195 – Attributes of Metering::ReadingQuality 167
Table 196 – Association ends of Metering::ReadingQuality with other classes 167
Table 197 – Attributes of Metering::ReadingQualityType 167
Table 198 – Association ends of Metering::ReadingQualityType with other classes 168
Table 199 – Attributes of Metering::ReadingType 168
Table 200 – Association ends of Metering::ReadingType with other classes 169
Table 201 – Attributes of Metering::Register 170
Table 202 – Association ends of Metering::Register with other classes 170
Table 203 – Attributes of Metering::ServiceMultiplier 170
Table 204 – Association ends of Metering::ServiceMultiplier with other classes 171
Table 205 – Attributes of Metering::SimpleEndDeviceFunction 171
Table 206 – Association ends of Metering::SimpleEndDeviceFunction with other classes 171
Table 207 – Attributes of Metering::UsagePoint 172
Table 208 – Association ends of Metering::UsagePoint with other classes 173
Table 209 – Attributes of Metering::UsagePointGroup 174
Table 210 – Association ends of Metering::UsagePointGroup with other classes 174
Table 211 – Attributes of Metering::UsagePointLocation 174
Table 212 – Association ends of Metering::UsagePointLocation with other classes 175
Table 213 – Attributes of LoadControl::RemoteConnectDisconnectInfo 176
Table 214 – Attributes of LoadControl::ConnectDisconnectFunction 177
Table 215 – Association ends of LoadControl::ConnectDisconnectFunction with other classes 177
Table 216 – Attributes of PaymentMetering::AccountMovement 184
Table 217 – Attributes of PaymentMetering::AccountingUnit 185
Table 218 – Attributes of PaymentMetering::BankAccountDetail 185
Table 219 – Attributes of PaymentMetering::Due 185
Table 220 – Attributes of PaymentMetering::LineDetail 186
Table 221 – Literals of PaymentMetering::ChargeKind 186
Table 222 – Literals of PaymentMetering::ChequeKind 186
Table 223 – Literals of PaymentMetering::SupplierKind 187
Table 224 – Literals of PaymentMetering::TenderKind 187
Table 225 – Literals of PaymentMetering::TransactionKind 187
Table 226 – Attributes of PaymentMetering::AuxiliaryAccount 188

Table 227 – Association ends of PaymentMetering::AuxiliaryAccount with other classes..... 188
Table 228 – Attributes of PaymentMetering::AuxiliaryAgreement.. 189
Table 229 – Association ends of PaymentMetering::AuxiliaryAgreement with other classes 190
Table 230 – Attributes of PaymentMetering::Card 190
Table 231 – Association ends of PaymentMetering::Card with other classes 190
Table 232 – Attributes of PaymentMetering::Cashier.. 191
Table 233 – Association ends of PaymentMetering::Cashier with other classes 191
Table 234 – Attributes of PaymentMetering::CashierShift.. 191
Table 235 – Association ends of PaymentMetering::CashierShift with other classes 192
Table 236 – Attributes of PaymentMetering::Charge 192
Table 237 – Association ends of PaymentMetering::Charge with other classes 192
Table 238 – Attributes of PaymentMetering::Cheque.. 193
Table 239 – Association ends of PaymentMetering::Cheque with other classes 193
Table 240 – Attributes of PaymentMetering::ConsumptionTariffInterval.............................. 193
Table 241 – Association ends of PaymentMetering::ConsumptionTariffInterval with other classes 194
Table 242 – Attributes of PaymentMetering::MerchantAccount.. 194
Table 243 – Association ends of PaymentMetering::MerchantAccount with other classes 194
Table 244 – Attributes of PaymentMetering::MerchantAgreement 195
Table 245 – Association ends of PaymentMetering::MerchantAgreement with other classes 195
Table 246 – Attributes of PaymentMetering::PointOfSale 196
Table 247 – Association ends of PaymentMetering::PointOfSale with other classes............ 196
Table 248 – Attributes of PaymentMetering::Receipt.. 196
Table 249 – Association ends of PaymentMetering::Receipt with other classes.................. 196
Table 250 – Attributes of PaymentMetering::ServiceSupplier 197
Table 251 – Association ends of PaymentMetering::ServiceSupplier with other classes 197
Table 252 – Attributes of PaymentMetering::Shift.. 198
Table 253 – Association ends of PaymentMetering::Shift with other classes 198
Table 254 – Attributes of PaymentMetering::TariffProfile.. 199
Table 255 – Association ends of PaymentMetering::TariffProfile with other classes 199
Table 256 – Attributes of PaymentMetering::Tender.. 200
Table 257 – Association ends of PaymentMetering::Tender with other classes 200
Table 258 – Attributes of PaymentMetering::TimeTariffInterval 200
Table 259 – Association ends of PaymentMetering::TimeTariffInterval with other classes 201
Table 260 – Attributes of PaymentMetering::Transaction 201
Table 261 – Association ends of PaymentMetering::Transaction with other classes 202
Table 262 – Attributes of PaymentMetering::Transactor 202
Table 263 – Association ends of PaymentMetering::Transactor with other classes.............. 202
Table 264 – Attributes of PaymentMetering::Vendor 203

Table 265 – Association ends of PaymentMetering::Vendor with other classes 203
Table 266 – Attributes of PaymentMetering::VendorShift 203
Table 267 – Association ends of PaymentMetering::VendorShift with other classes 204

INTERNATIONAL ELECTROTECHNICAL COMMISSION

____

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

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

FOREWORD

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

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

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

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

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

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

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

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

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

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

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

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

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

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

INTRODUCTION

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

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

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

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

1 Scope

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

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

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

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

2 Normative references

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

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

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

[3] Under consideration.

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

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

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

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

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

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

3 Terms and definitions

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

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

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

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

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

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

4 5th edition. Under consideration.

5 Under consideration.

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

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

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

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

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

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

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

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

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

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

4 CIM specification

4.1 CIM modelling notation

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

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

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

4.2 CIM packages

4.2.1 General

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

4.2.2 CIM packages

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

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

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

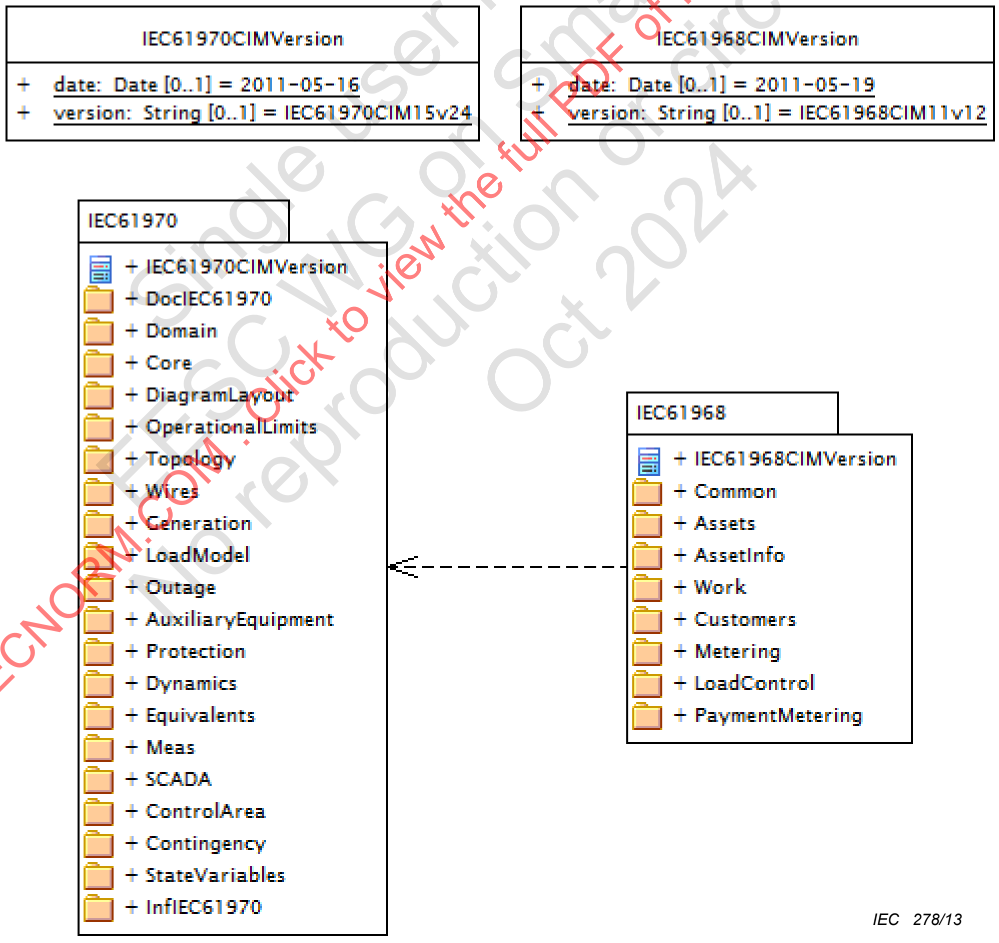


Figure 1 – CIM packages

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

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

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

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

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

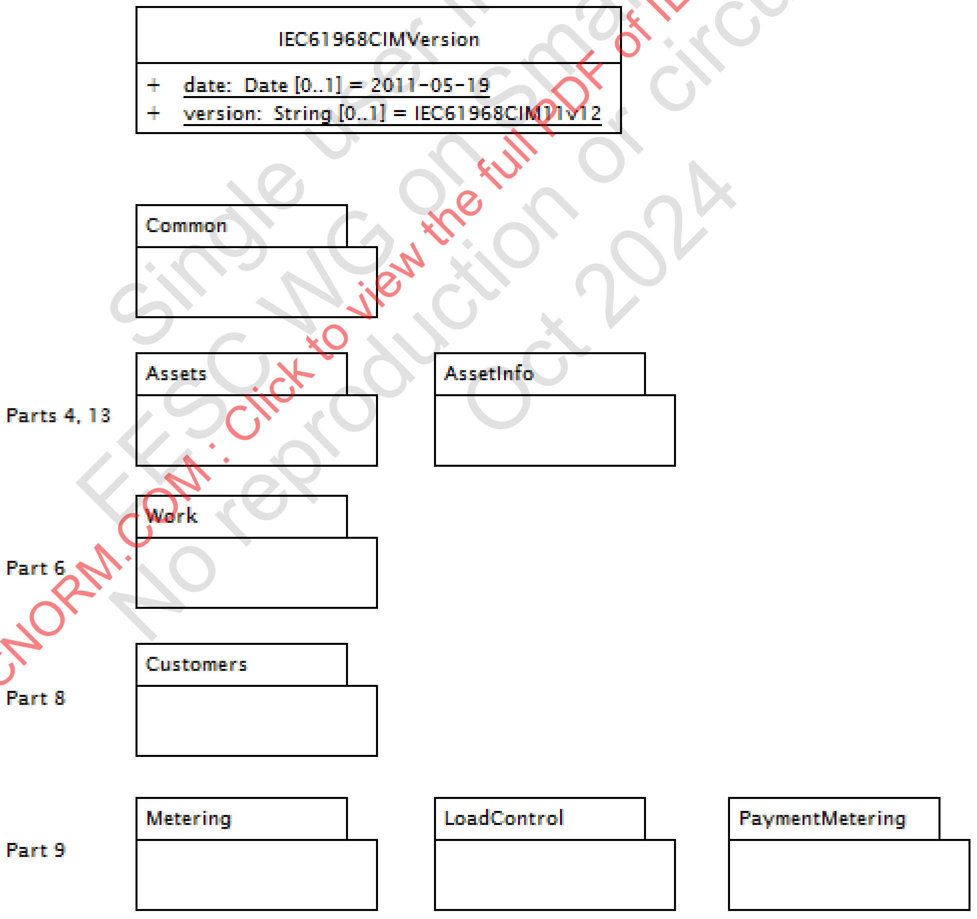


Figure 2 – CIM extensions for distribution (DCIM) top-level packages

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

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

4.3 CIM UML modelling

4.3.1 General

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

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

4.3.2 Scope of UML model

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

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

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

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

4.3.3 Extensibility

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

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

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

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

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

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

4.3.4 Message payload (profile) definition

4.3.4.1 General

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

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

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

4.3.4.2 XML schema syntax

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

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

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

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

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

4.3.4.3 RDF schema syntax

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

8 Under consideration.

4.3.4.4 Data exchange contexts supported by DCIM

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

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

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

4.4 DCIM model concepts and examples

4.4.1 General

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

4.4.2 Common concepts

4.4.2.1 Key classes in DCIM

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

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

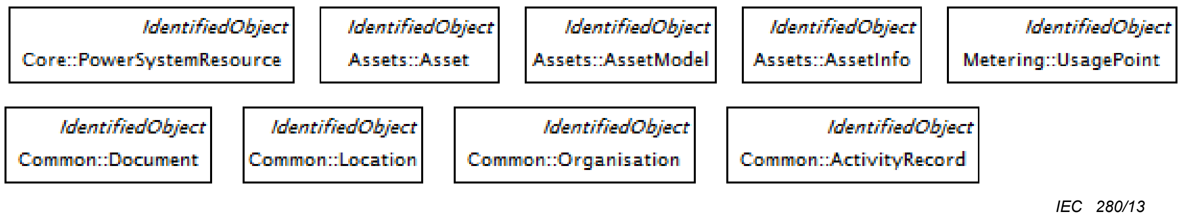


Figure 3 – DCIM key classes

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

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

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

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

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

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

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

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

4.4.2.2 Locations and geographical representations

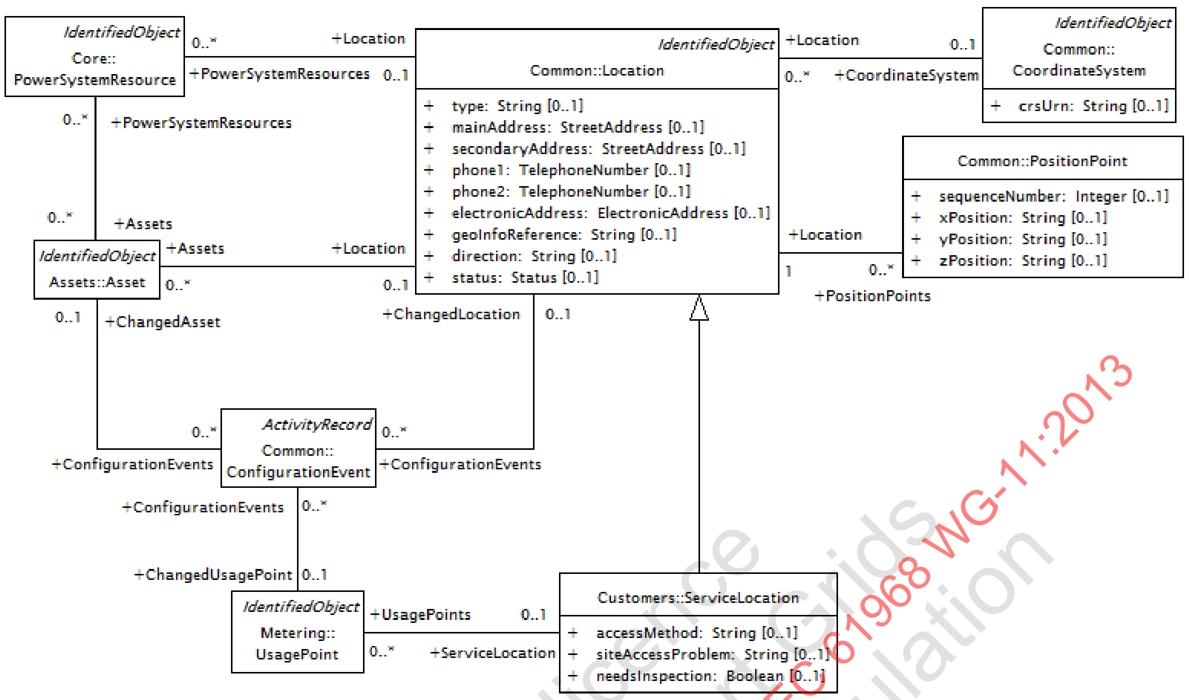
To allow for locating different entities and activities in space, DCIM provides the class **Location**. Entities and activities can be located by their different kinds of addresses as well as by position coordinates, **PositionPoint** in a given **CoordinateSystem**. Changes in configuration of a **Location**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

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

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

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

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



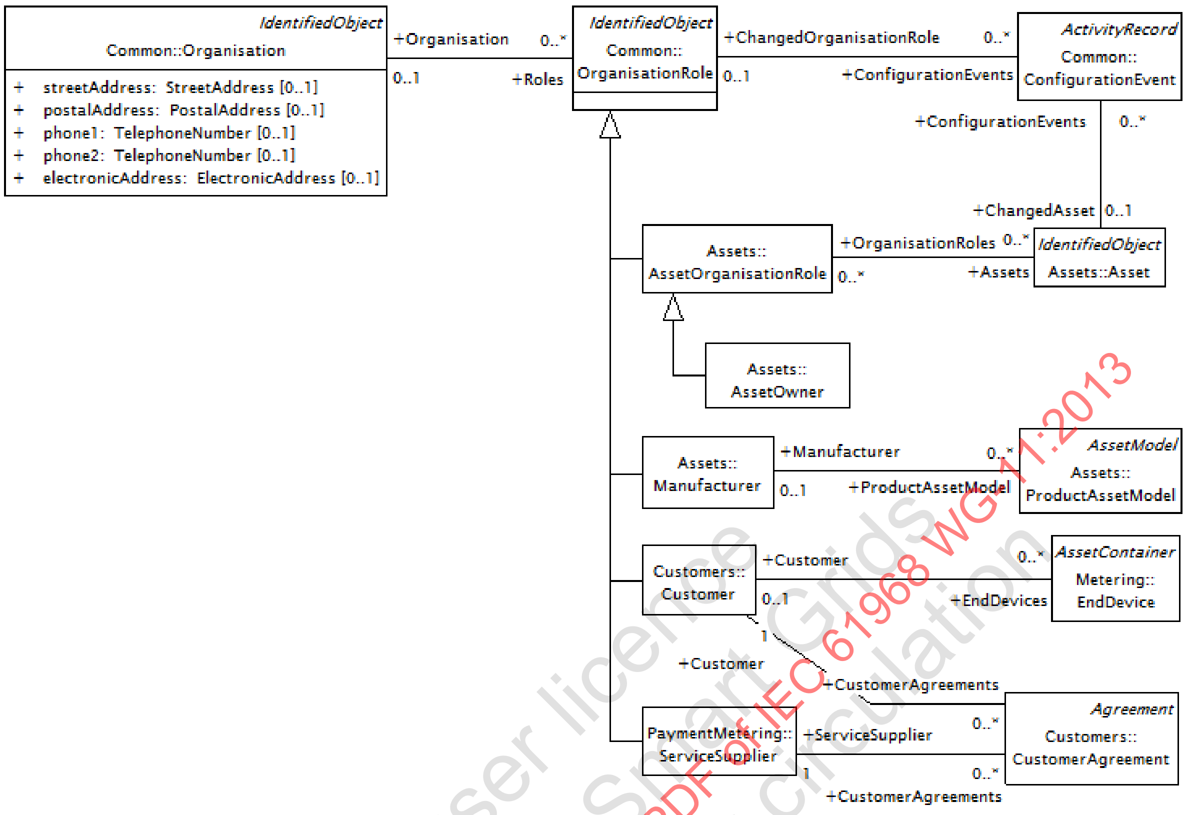
IEC 281/13

Figure 4 – DCIM power system resource and asset locations

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

4.4.2.3 Organisations

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



IEC 282/13

Figure 5 – DCIM organization model

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

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

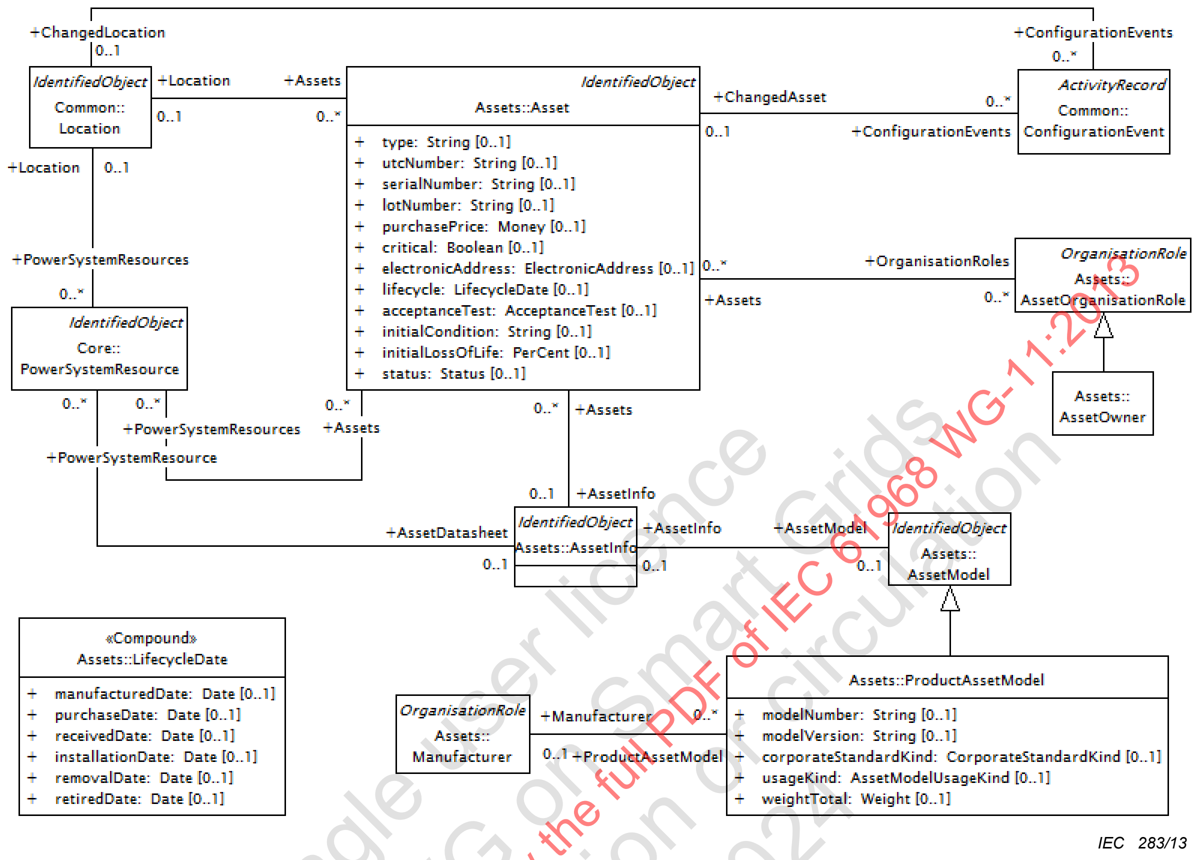
In DCIM, **Organisation** as a business entity is never specialised.

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

4.4.2.4 Assets

4.4.2.4.1 Major classes in assets model

Whereas IEC 61968-11:2010 (1st edition) contained **Asset** and **AssetModel** classes required only in the context of metering (IEC 61968-9), this part of IEC 61968 (2nd edition) provides a more elaborate assets model, supporting the needs for data exchanges in the context of network model exchanges (IEC 61968-4 and IEC 61968-13). Figure 6 gives an overview of major classes that are included in the DCIM assets model.



+ChangedLocation
0..1
+ConfigurationEvents
0..*
IdentifiedObject
Common::Location
+Location
0..1
+Assets
0..*
IdentifiedObject
Assets::Asset
+ type: String [0..1]
+ utcNumber: String [0..1]
+ serialNumber: String [0..1]
+ lotNumber: String [0..1]
+ purchasePrice: Money [0..1]
+ critical: Boolean [0..1]
+ electronicAddress: ElectronicAddress [0..1]
+ lifecycle: LifecycleDate [0..1]
+ acceptanceTest: AcceptanceTest [0..1]
+ initialCondition: String [0..1]
+ initialLossOfLife: PerCent [0..1]
+ status: Status [0..1]
+ChangedAsset
0..1
+ConfigurationEvents
0..*
ActivityRecord
Common::ConfigurationEvent
+Location
0..1
+PowerSystemResources
0..*
IdentifiedObject
Core::PowerSystemResource
0..*
+OrganisationRoles
+Assets
0..*
OrganisationRole
Assets::AssetOrganisationRole
Assets::AssetOwner
0..*
0..*
+PowerSystemResources
0..*
+Assets
+PowerSystemResource
0..*
+Assets
0..1
+AssetInfo
+AssetDatasheet
0..1
IdentifiedObject
Assets::AssetInfo
+AssetInfo
0..1
+AssetModel
0..1
IdentifiedObject
Assets::AssetModel
«Compound»
Assets::LifecycleDate
+ manufacturedDate: Date [0..1]
+ purchaseDate: Date [0..1]
+ receivedDate: Date [0..1]
+ installationDate: Date [0..1]
+ removalDate: Date [0..1]
+ retiredDate: Date [0..1]
OrganisationRole
Assets::Manufacturer
+Manufacturer
0..*
0..1
+ProductAssetModel
Assets::ProductAssetModel
+ modelNumber: String [0..1]
+ modelVersion: String [0..1]
+ corporateStandardKind: CorporateStandardKind [0..1]
+ usageKind: AssetModelUsageKind [0..1]
+ weightTotal: Weight [0..1]
IEC 283/13

Figure 6 – DCIM assets model

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

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

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

Another essential class in the assets model is **AssetInfo**, which is an identifiable container for various physical parameters describing associated **Asset**s. An **Asset** without the associated concrete **AssetInfo** instance is not fully described. **AssetInfo**, modelling what is sometimes called “catalogue”, “library” or “code”, is a set of attributes of an asset, representing typical datasheet information of a physical device that can be instantiated and shared in different data exchange contexts:

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

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

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

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

4.4.2.4.2 Specialisations in assets model

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

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

a) Among classes **Asset**, **AssetModel**, **ProductAssetModel** and **AssetInfo**: **ProductAssetModel** should never be specialised in the standard DCIM, and **AssetModel** is specialised only with **ProductAssetModel**[11]. **AssetInfo** will always be specialised. **Asset** may be specialised according to guideline d) only.
b) If a device/equipment exists in **IEC61970::Wires** model package of IEC 61970-301 (as a specialisation of, or related to **PowerSystemResource**), it is not allowed to define its counterpart as a specialisation of **Asset**. In contrast, a specialisation of **AssetInfo** shall be defined with its datasheet properties, where applicable.
c) Relationship between functional and physical models is established between relevant specialisations of **PowerSystemResource** and **AssetInfo** through inherited relationship,

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

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

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

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

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

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

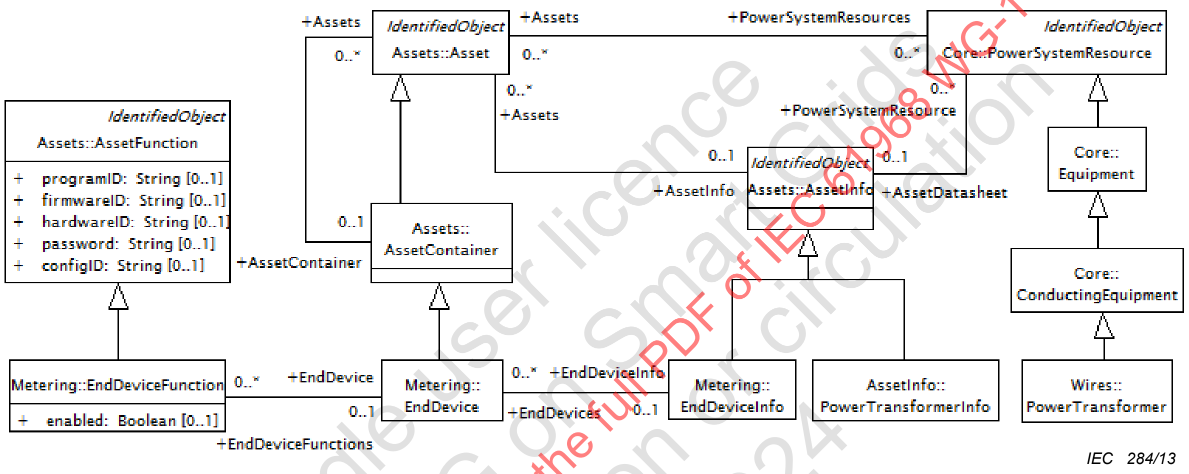


Figure 7 – DCIM assets – specialising guidelines

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

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

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

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

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

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

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

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

4.4.2.5 Electrical model vs. physical model

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

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

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

4.4.3 Network modelling concepts and examples

4.4.3.1 Connectivity and phase modelling

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

For an extensive example of balanced three-phase sample network, refer to IEC 61970-301:—, 4.4.4.2, Connectivity and containment example, and for an example illustrating how the CIM connectivity model can be used to represent a multi-phase network, refer to IEC 61970-301:—, 4.4.8, Phase wire modelling.

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

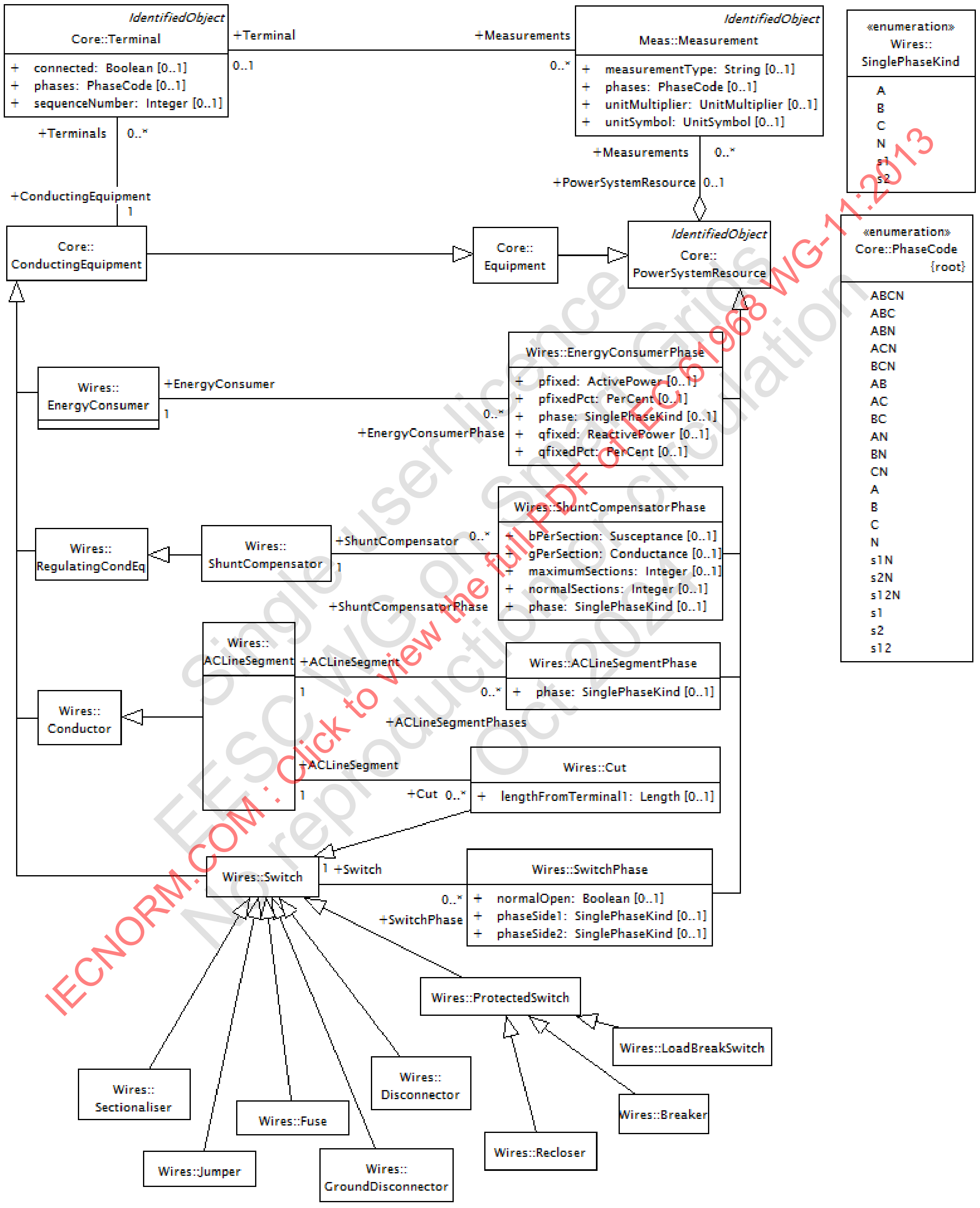

IdentifiedObject
Core::Terminal
+ connected: Boolean [0..1]
+ phases: PhaseCode [0..1]
+ sequenceNumber: Integer [0..1]
+Terminal
0..1
+Measurements
0..*
IdentifiedObject
Meas::Measurement
+ measurementType: String [0..1]
+ phases: PhaseCode [0..1]
+ unitMultiplier: UnitMultiplier [0..1]
+ unitSymbol: UnitSymbol [0..1]
«enumeration» Wires:: SinglePhaseKind
A
B
C
N
s1
s2
+Terminals 0..*
+ConductingEquipment 1
+Measurements 0..*
+PowerSystemResource 0..1
Core:: ConductingEquipment
Core:: Equipment
IdentifiedObject
Core:: PowerSystemResource
«enumeration» Core::PhaseCode {root}
ABCN
ABC
ABN
ACN
BCN
AB
AC
BC
AN
BN
CN
A
B
C
N
s1N
s2N
s12N
s1
s2
s12
Wires::EnergyConsumerPhase
+ pfixed: ActivePower [0..1]
+ pfixedPct: PerCent [0..1]
+ phase: SinglePhaseKind [0..1]
+ qfixed: ReactivePower [0..1]
+ qfixedPct: PerCent [0..1]
Wires:: EnergyConsumer
+EnergyConsumer 1
+EnergyConsumerPhase 0..*
Wires::ShuntCompensatorPhase
+ bPerSection: Susceptance [0..1]
+ gPerSection: Conductance [0..1]
+ maximumSections: Integer [0..1]
+ normalSections: Integer [0..1]
+ phase: SinglePhaseKind [0..1]
Wires:: RegulatingCondEq
Wires:: ShuntCompensator
+ShuntCompensator 1
+ShuntCompensatorPhase 0..*
Wires:: ACLineSegment
+ACLineSegment 1
Wires::ACLineSegmentPhase
+ phase: SinglePhaseKind [0..1]
+ACLineSegmentPhases 0..*
Wires:: Conductor
+ACLineSegment 1
+Cut 0..*
Wires::Cut
+ lengthFromTerminal1: Length [0..1]
Wires::Switch
1 +Switch
Wires::SwitchPhase
+ normalOpen: Boolean [0..1]
+ phaseSide1: SinglePhaseKind [0..1]
+ phaseSide2: SinglePhaseKind [0..1]
0..* +SwitchPhase
Wires::ProtectedSwitch
Wires::LoadBreakSwitch
Wires::Breaker
Wires::Recloser
Wires:: Sectionaliser
Wires::Fuse
Wires:: Disconnector
Wires::Jumper
Wires:: GroundDisconnector
IEC 285/13

Figure 8 – DCIM phase modelling

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

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

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

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

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

4.4.3.2 Single-phase and unbalanced loads

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

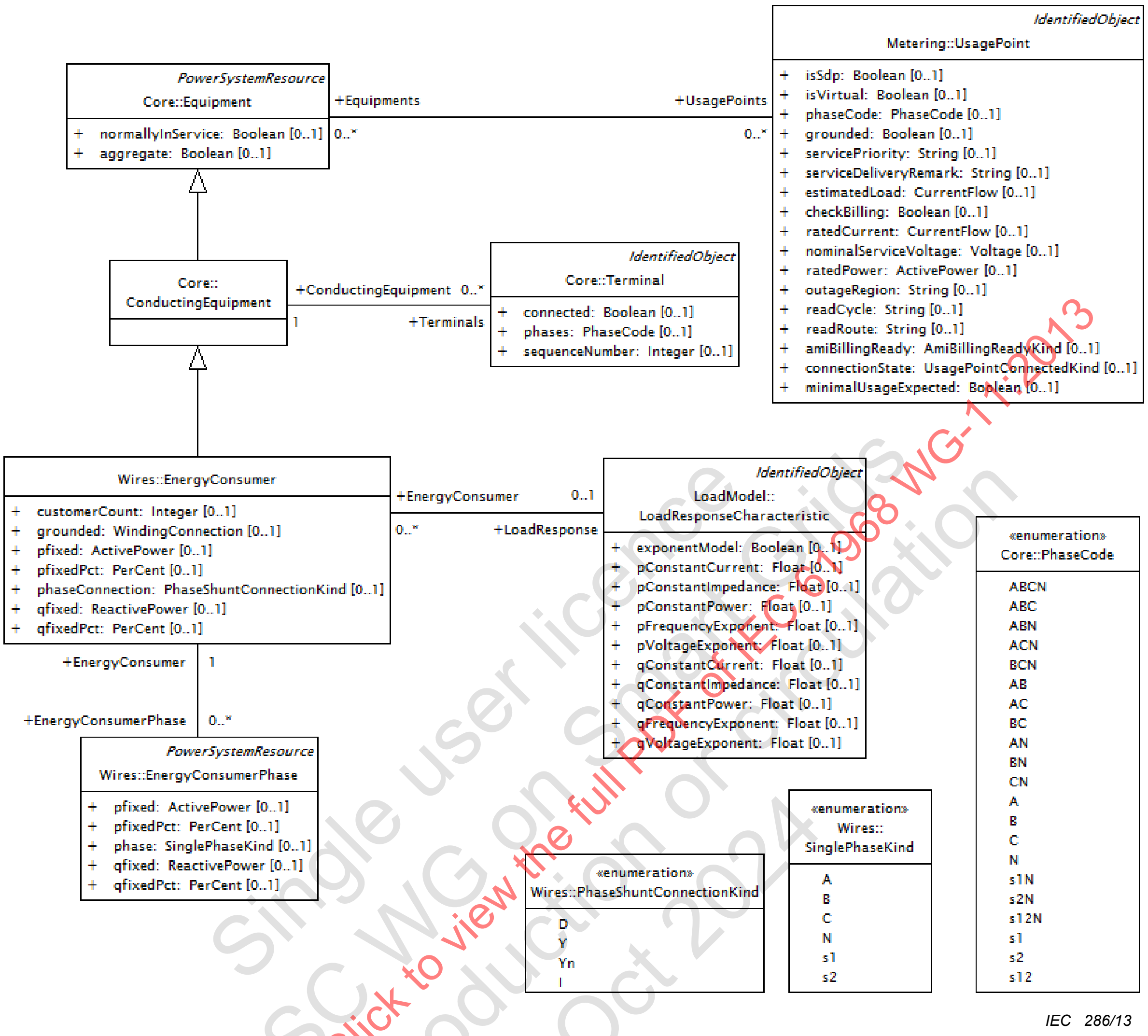

IdentifiedObject
Metering::UsagePoint
+ isSdp: Boolean [0..1]
+ isVirtual: Boolean [0..1]
+ phaseCode: PhaseCode [0..1]
+ grounded: Boolean [0..1]
+ servicePriority: String [0..1]
+ serviceDeliveryRemark: String [0..1]
+ estimatedLoad: CurrentFlow [0..1]
+ checkBilling: Boolean [0..1]
+ ratedCurrent: CurrentFlow [0..1]
+ nominalServiceVoltage: Voltage [0..1]
+ ratedPower: ActivePower [0..1]
+ outageRegion: String [0..1]
+ readCycle: String [0..1]
+ readRoute: String [0..1]
+ amiBillingReady: AmiBillingReadyKind [0..1]
+ connectionState: UsagePointConnectedKind [0..1]
+ minimalUsageExpected: Boolean [0..1]
PowerSystemResource
Core::Equipment
+ normallyInService: Boolean [0..1]
+ aggregate: Boolean [0..1]
+Equipments
0..*
+UsagePoints
0..*
Core::
ConductingEquipment
+ConductingEquipment 0..*
1
+Terminals
IdentifiedObject
Core::Terminal
+ connected: Boolean [0..1]
+ phases: PhaseCode [0..1]
+ sequenceNumber: Integer [0..1]
Wires::EnergyConsumer
+ customerCount: Integer [0..1]
+ grounded: WindingConnection [0..1]
+ pfixed: ActivePower [0..1]
+ pfixedPct: PerCent [0..1]
+ phaseConnection: PhaseShuntConnectionKind [0..1]
+ qfixed: ReactivePower [0..1]
+ qfixedPct: PerCent [0..1]
+EnergyConsumer
0..1
0..*
+LoadResponse
IdentifiedObject
LoadModel::
LoadResponseCharacteristic
+ exponentModel: Boolean [0..1]
+ pConstantCurrent: Float [0..1]
+ pConstantImpedance: Float [0..1]
+ pConstantPower: Float [0..1]
+ pFrequencyExponent: Float [0..1]
+ pVoltageExponent: Float [0..1]
+ qConstantCurrent: Float [0..1]
+ qConstantImpedance: Float [0..1]
+ qConstantPower: Float [0..1]
+ qFrequencyExponent: Float [0..1]
+ qVoltageExponent: Float [0..1]
«enumeration»
Core::PhaseCode
ABCN
ABC
ABN
ACN
BCN
AB
AC
BC
AN
BN
CN
A
B
C
N
s1N
s2N
s12N
s1
s2
s12
+EnergyConsumer
1
+EnergyConsumerPhase
0..*
PowerSystemResource
Wires::EnergyConsumerPhase
+ pfixed: ActivePower [0..1]
+ pfixedPct: PerCent [0..1]
+ phase: SinglePhaseKind [0..1]
+ qfixed: ReactivePower [0..1]
+ qfixedPct: PerCent [0..1]
«enumeration»
Wires::PhaseShuntConnectionKind
D
Y
Yn
I
«enumeration»
Wires::
SinglePhaseKind
A
B
C
N
s1
s2
IEC 286/13

Figure 9 – DCIM load model

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

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

If a three-phase load is unbalanced, it can still be modelled with the same **EnergyConsumer** instance, now referring to three additional **EnergyConsumerPhase** instances, each representing one phase with its **phase** attribute. This makes it possible to use an **EnergyConsumer** with its identity for unbalanced modelling as well. With single-phase or two-phase loads, the model should also include correctly phased conductors or transformers that establish connectivity back to the source. The total real and reactive power need not be specified on **EnergyConsumer** attributes, because they can be derived from the distribution among phases specified on **EnergyConsumerPhase** attributes. Each **phase** can be **A**, **B**, **C**, **S1**, or **S2** (**N** does not apply to this case). For a delta or phase-to-phase connected load, which is indicated with **D** for **phaseConnection**:

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

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

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

4.4.3.3 Distribution line segments

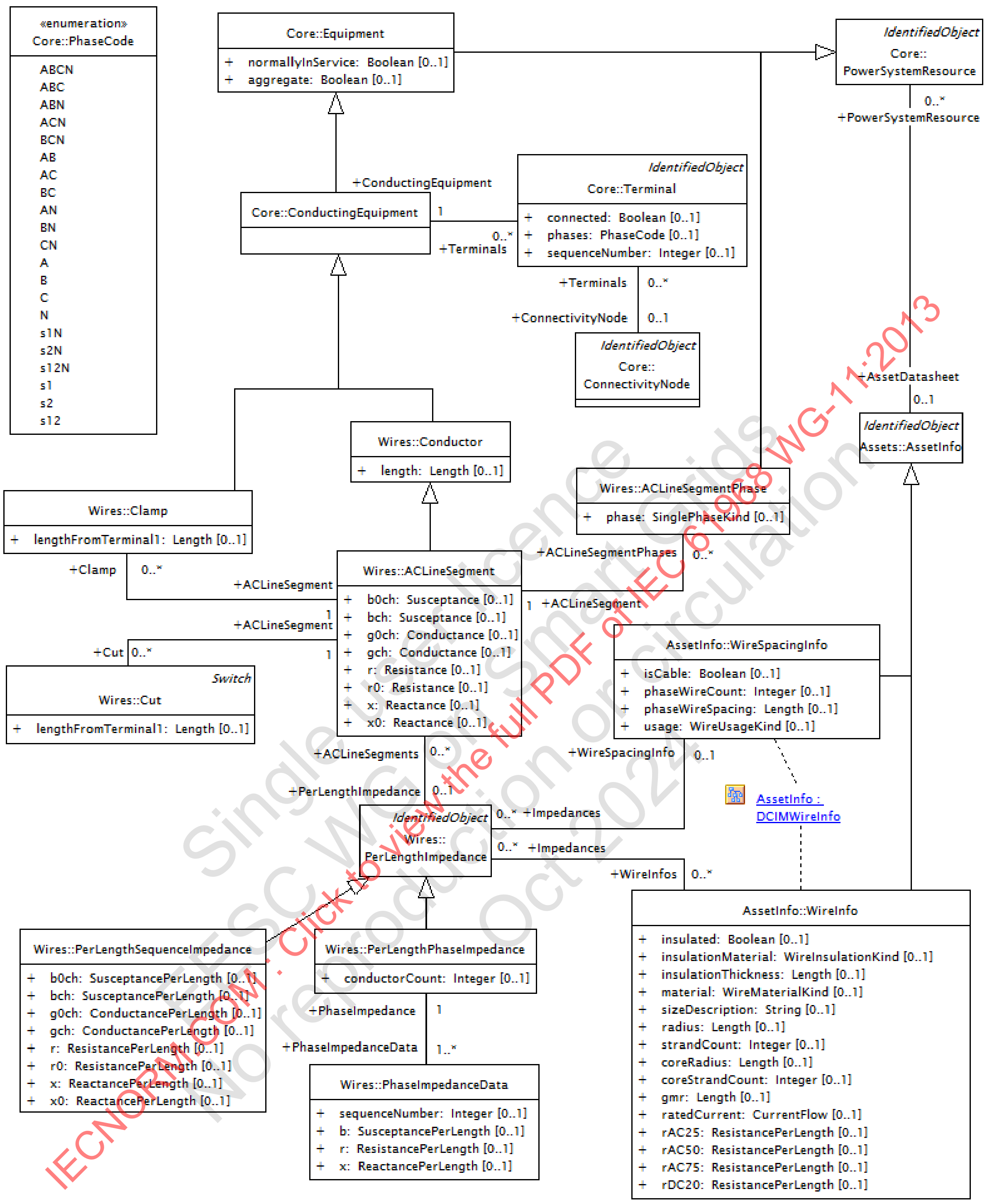
Figure 10 shows the classes available to model AC line segments (i.e., conductors). There are three ways to describe the impedance parameters of an **ACLineSegment** using only the **Wires** package, and a fourth way making use of the **AssetInfo** package.

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

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

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

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


«enumeration» Core::PhaseCode
ABCN
ABC
ABN
ACN
BCN
AB
AC
BC
AN
BN
CN
A
B
C
N
s1N
s2N
s12N
s1
s2
s12
Core::Equipment
+ normallyInService: Boolean [0..1]
+ aggregate: Boolean [0..1]
IdentifiedObject
Core::PowerSystemResource
0..*
+PowerSystemResource
+ConductingEquipment
Core::ConductingEquipment
1
0..*
+Terminals
IdentifiedObject
Core::Terminal
+ connected: Boolean [0..1]
+ phases: PhaseCode [0..1]
+ sequenceNumber: Integer [0..1]
+Terminals
0..*
+ConnectivityNode
0..1
IdentifiedObject
Core::ConnectivityNode
+AssetDatasheet
0..1
IdentifiedObject
Assets::AssetInfo
Wires::Conductor
+ length: Length [0..1]
Wires::Clamp
+ lengthFromTerminal1: Length [0..1]
+Clamp
0..*
+ACLineSegment
Wires::ACLineSegmentPhase
+ phase: SinglePhaseKind [0..1]
+ACLineSegmentPhases
0..*
Wires::ACLineSegment
+ b0ch: Susceptance [0..1]
+ bch: Susceptance [0..1]
+ g0ch: Conductance [0..1]
+ gch: Conductance [0..1]
+ r: Resistance [0..1]
+ r0: Resistance [0..1]
+ x: Reactance [0..1]
+ x0: Reactance [0..1]
1
+ACLineSegment
+ACLineSegment
1
+Cut
0..*
1
Switch
Wires::Cut
+ lengthFromTerminal1: Length [0..1]
AssetInfo::WireSpacingInfo
+ isCable: Boolean [0..1]
+ phaseWireCount: Integer [0..1]
+ phaseWireSpacing: Length [0..1]
+ usage: WireUsageKind [0..1]
+ACLineSegments
0..*
+WireSpacingInfo
0..1
+PerLengthImpedance
0..1
IdentifiedObject
Wires::PerLengthImpedance
0..*
+Impedances
0..*
+Impedances
AssetInfo : DCIMWireInfo
+WireInfos
0..*
AssetInfo::WireInfo
+ insulated: Boolean [0..1]
+ insulationMaterial: WireInsulationKind [0..1]
+ insulationThickness: Length [0..1]
+ material: WireMaterialKind [0..1]
+ sizeDescription: String [0..1]
+ radius: Length [0..1]
+ strandCount: Integer [0..1]
+ coreRadius: Length [0..1]
+ coreStrandCount: Integer [0..1]
+ gmr: Length [0..1]
+ ratedCurrent: CurrentFlow [0..1]
+ rAC25: ResistancePerLength [0..1]
+ rAC50: ResistancePerLength [0..1]
+ rAC75: ResistancePerLength [0..1]
+ rDC20: ResistancePerLength [0..1]
Wires::PerLengthSequenceImpedance
+ b0ch: SusceptancePerLength [0..1]
+ bch: SusceptancePerLength [0..1]
+ g0ch: ConductancePerLength [0..1]
+ gch: ConductancePerLength [0..1]
+ r: ResistancePerLength [0..1]
+ r0: ResistancePerLength [0..1]
+ x: ReactancePerLength [0..1]
+ x0: ReactancePerLength [0..1]
Wires::PerLengthPhaseImpedance
+ conductorCount: Integer [0..1]
+PhaseImpedance
1
+PhaseImpedanceData
1..*
Wires::PhaseImpedanceData
+ sequenceNumber: Integer [0..1]
+ b: SusceptancePerLength [0..1]
+ r: ResistancePerLength [0..1]
+ x: ReactancePerLength [0..1]

IEC 287/13

Figure 10 – DCIM line connectivity model

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

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

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

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

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

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

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

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

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

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

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

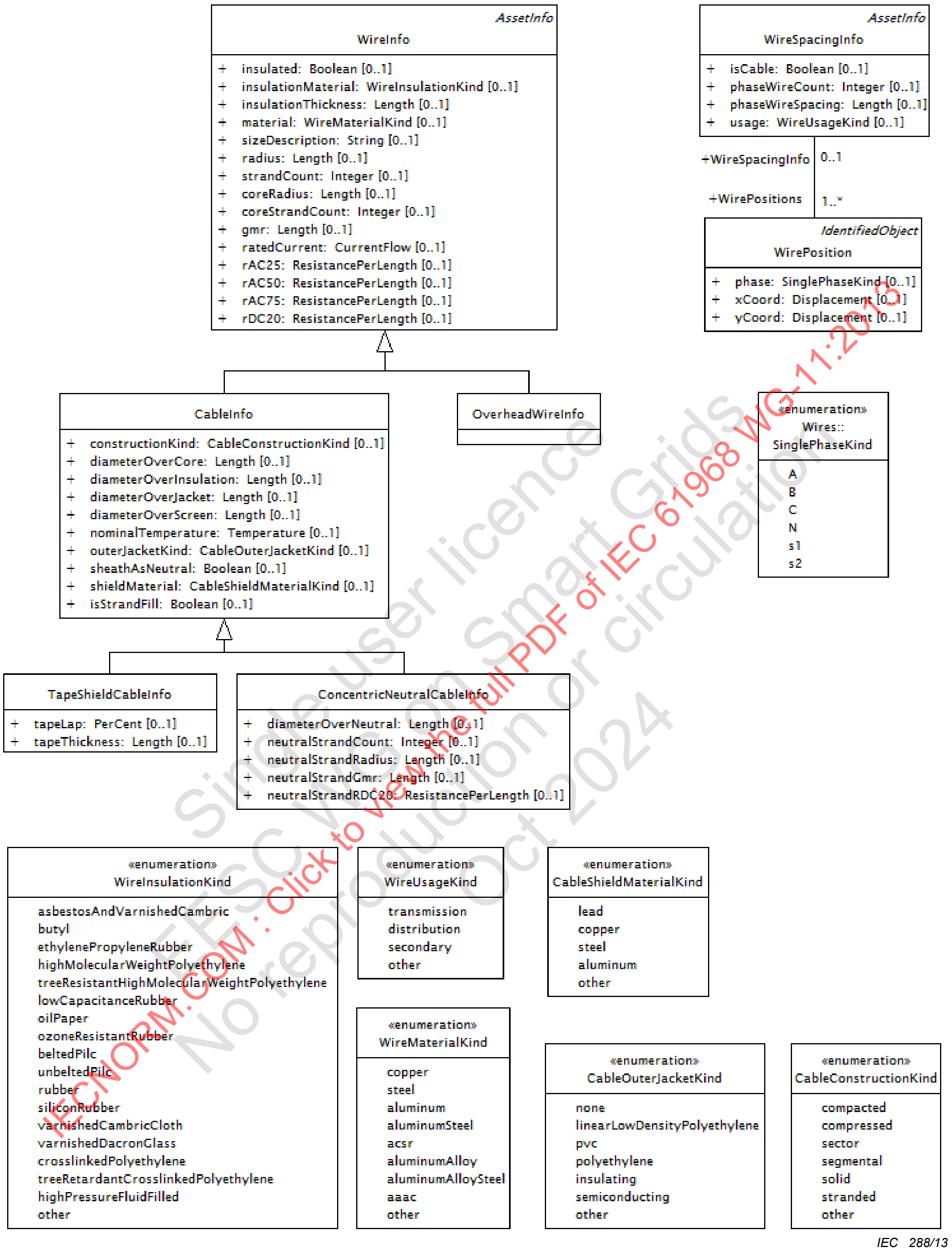

AssetInfo
WireInfo
+ insulated: Boolean [0..1]
+ insulationMaterial: WireInsulationKind [0..1]
+ insulationThickness: Length [0..1]
+ material: WireMaterialKind [0..1]
+ sizeDescription: String [0..1]
+ radius: Length [0..1]
+ strandCount: Integer [0..1]
+ coreRadius: Length [0..1]
+ coreStrandCount: Integer [0..1]
+ gmr: Length [0..1]
+ ratedCurrent: CurrentFlow [0..1]
+ rAC25: ResistancePerLength [0..1]
+ rAC50: ResistancePerLength [0..1]
+ rAC75: ResistancePerLength [0..1]
+ rDC20: ResistancePerLength [0..1]
AssetInfo
WireSpacingInfo
+ isCable: Boolean [0..1]
+ phaseWireCount: Integer [0..1]
+ phaseWireSpacing: Length [0..1]
+ usage: WireUsageKind [0..1]
+WireSpacingInfo
0..1
+WirePositions
1..*
IdentifiedObject
WirePosition
+ phase: SinglePhaseKind [0..1]
+ xCoord: Displacement [0..1]
+ yCoord: Displacement [0..1]
CableInfo
+ constructionKind: CableConstructionKind [0..1]
+ diameterOverCore: Length [0..1]
+ diameterOverInsulation: Length [0..1]
+ diameterOverJacket: Length [0..1]
+ diameterOverScreen: Length [0..1]
+ nominalTemperature: Temperature [0..1]
+ outerJacketKind: CableOuterJacketKind [0..1]
+ sheathAsNeutral: Boolean [0..1]
+ shieldMaterial: CableShieldMaterialKind [0..1]
+ isStrandFill: Boolean [0..1]
OverheadWireInfo
«enumeration»
Wires::
SinglePhaseKind
A
B
C
N
s1
s2
TapeShieldCableInfo
+ tapeLap: PerCent [0..1]
+ tapeThickness: Length [0..1]
ConcentricNeutralCableInfo
+ diameterOverNeutral: Length [0..1]
+ neutralStrandCount: Integer [0..1]
+ neutralStrandRadius: Length [0..1]
+ neutralStrandGmr: Length [0..1]
+ neutralStrandRDC20: ResistancePerLength [0..1]
«enumeration»
WireInsulationKind
asbestosAndVarnishedCambric
butyl
ethylenePropyleneRubber
highMolecularWeightPolyethylene
treeResistantHighMolecularWeightPolyethylene
lowCapacitanceRubber
oilPaper
ozoneResistantRubber
beltedPilc
unbeltedPilc
rubber
siliconRubber
varnishedCambricCloth
varnishedDacronGlass
crosslinkedPolyethylene
treeRetardantCrosslinkedPolyethylene
highPressureFluidFilled
other
«enumeration»
WireUsageKind
transmission
distribution
secondary
other
«enumeration»
CableShieldMaterialKind
lead
copper
steel
aluminum
other
«enumeration»
WireMaterialKind
copper
steel
aluminum
aluminumSteel
acsr
aluminumAlloy
aluminumAlloySteel
aaac
other
«enumeration»
CableOuterJacketKind
none
linearLowDensityPolyethylene
pvc
polyethylene
insulating
semiconducting
other
«enumeration»
CableConstructionKind
compacted
compressed
sector
segmental
solid
stranded
other
IEC 288/13

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

WireInfo attributes **material**, **sizeDescription**, **strandCount**, and **coreStrandCount** (for ACSR, aluminium conductor steel reinforced wire) help to identify the wire, and are often coded into the instance local name. The minimum required electrical attributes are **gmr**, **radius**, and **rAC50**. A complete wire table would usually have resistances defined at other temperatures and frequencies, such as **rAC25**, **rAC75**, and **rDC20**. For ACSR wires, the **coreRadius** is optional for frequency-dependent calculation of the **gmr.** The **coreRadius** is zero for non-ACSR wires. The **ratedCurrent** is the ampacity at 50 ˚C. The **ratedCurrent** is necessary to interpret power flow output, but not for the power flow solution itself. The **insulated**, **insulationMaterial**, and **insulationThickness** attributes apply mainly to the derived cable classes, but they also support triplex secondary lines and overhead spacer cable. The **CableInfo** attributes describe additional properties of layers over the conducting core. The **diameterOverCore** includes both conductor and semi-conducting screen; it should be the insulation's inside diameter. The **diameterOverInsulation** is the insulating layer's outside diameter, not including any outside screens or semi-conducting layers. The **diameterOverScreen** includes the insulation plus screen or semi-conducting layer; it should be the shield's or sheath's inside diameter. The **diameterOverJacket** is the cable's outside diameter; it should be the largest diameter value specified except for concentric neutrals.

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

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

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

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

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

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

4.4.3.3.1 Using sequence impedances (balanced case)

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

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

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

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

$$Z_1 = Z_0 = Z_s$$

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

$$Z_1 = Z_s - Z_m$$

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

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

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

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

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

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

4.4.3.3.2 Using phase impedances (unbalanced case)

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

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

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

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

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

4.4.3.3.3 Using physical parameters

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

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

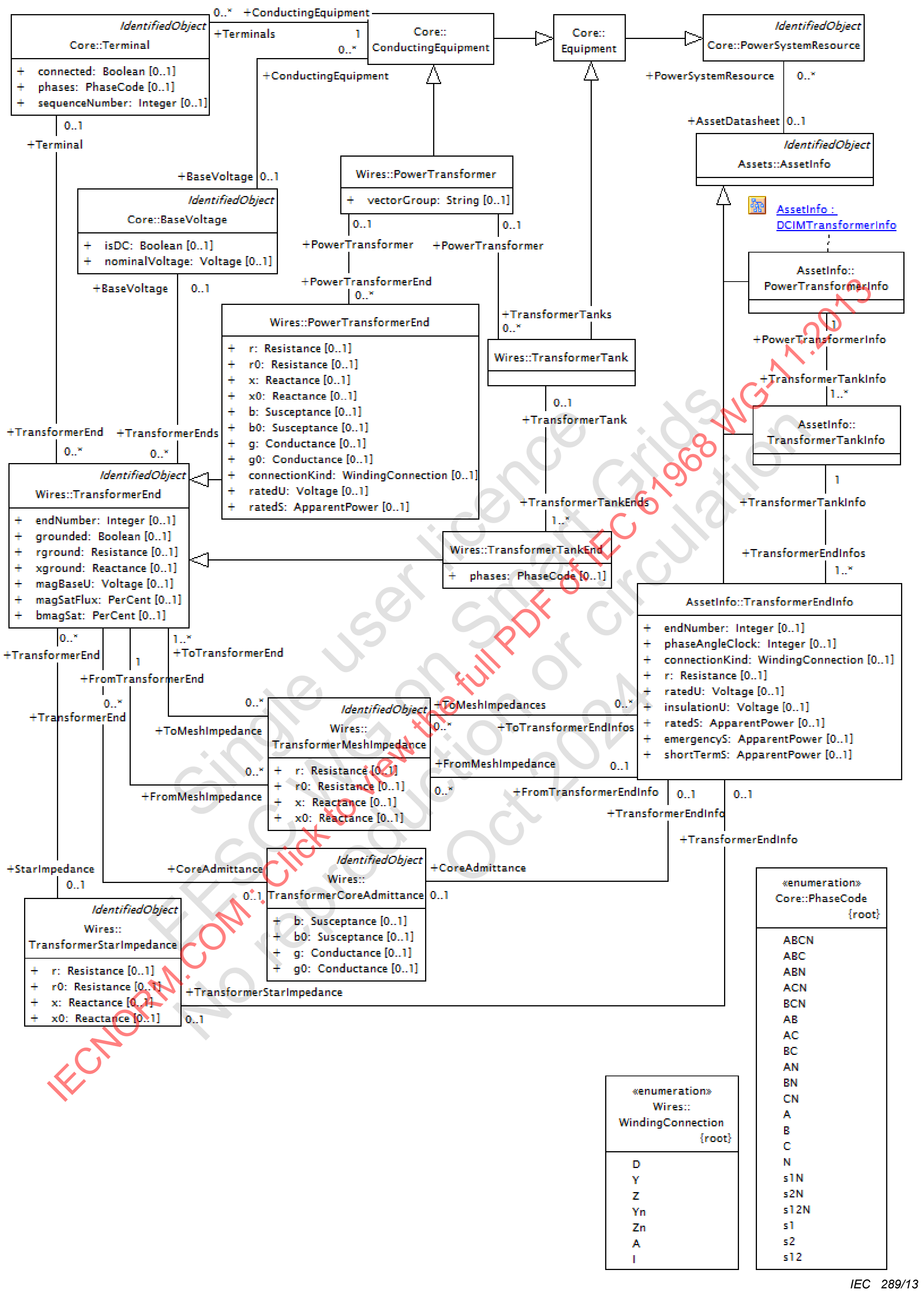
A single-phase, concentric neutral cable requires one instance of **ConcentricNeutralCableInfo**, one **WireSpacingInfo** with attribute **isCable**=true, and one **WirePosition** instance to specify the burial depth. A three-phase tape shielded cable, with bare neutral conductor, would require one instance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

4.4.3.4 Distribution transformers

4.4.3.4.1 Electrical model

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

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



IdentifiedObject
Core::Terminal
+ connected: Boolean [0..1]
+ phases: PhaseCode [0..1]
+ sequenceNumber: Integer [0..1]
0..* +ConductingEquipment
+Terminals 1
0..*
Core::ConductingEquipment
Core::Equipment
IdentifiedObject
Core::PowerSystemResource
+PowerSystemResource 0..*
+AssetDatasheet 0..1
IdentifiedObject
Assets::AssetInfo
AssetInfo : DCIMTransformerInfo
+ConductingEquipment
0..1
+Terminal
+BaseVoltage 0..1
IdentifiedObject
Core::BaseVoltage
+ isDC: Boolean [0..1]
+ nominalVoltage: Voltage [0..1]
+BaseVoltage 0..1
Wires::PowerTransformer
+ vectorGroup: String [0..1]
0..1 0..1
+PowerTransformer +PowerTransformer
+PowerTransformerEnd 0..*
Wires::PowerTransformerEnd
+ r: Resistance [0..1]
+ r0: Resistance [0..1]
+ x: Reactance [0..1]
+ x0: Reactance [0..1]
+ b: Susceptance [0..1]
+ b0: Susceptance [0..1]
+ g: Conductance [0..1]
+ g0: Conductance [0..1]
+ connectionKind: WindingConnection [0..1]
+ ratedU: Voltage [0..1]
+ ratedS: ApparentPower [0..1]
+TransformerTanks 0..*
Wires::TransformerTank
0..1
+TransformerTank
AssetInfo::PowerTransformerInfo
1
+PowerTransformerInfo
+TransformerTankInfo 1..*
AssetInfo::TransformerTankInfo
1
+TransformerTankInfo
+TransformerEndInfos 1..*
+TransformerEnd 0..*
+TransformerEnds 0..*
IdentifiedObject
Wires::TransformerEnd
+ endNumber: Integer [0..1]
+ grounded: Boolean [0..1]
+ rground: Resistance [0..1]
+ xground: Reactance [0..1]
+ magBaseU: Voltage [0..1]
+ magSatFlux: PerCent [0..1]
+ bmagSat: PerCent [0..1]
+TransformerTankEnds 1..*
Wires::TransformerTankEnd
+ phases: PhaseCode [0..1]
AssetInfo::TransformerEndInfo
+ endNumber: Integer [0..1]
+ phaseAngleClock: Integer [0..1]
+ connectionKind: WindingConnection [0..1]
+ r: Resistance [0..1]
+ ratedU: Voltage [0..1]
+ insulationU: Voltage [0..1]
+ ratedS: ApparentPower [0..1]
+ emergencyS: ApparentPower [0..1]
+ shortTermS: ApparentPower [0..1]
0..* 1..*
+TransformerEnd +ToTransformerEnd
1
+FromTransformerEnd
0..*
+TransformerEnd
0..*
+ToMeshImpedance
IdentifiedObject
Wires::TransformerMeshImpedance
+ r: Resistance [0..1]
+ r0: Resistance [0..1]
+ x: Reactance [0..1]
+ x0: Reactance [0..1]
+ToMeshImpedances 0..*
0..* +ToTransformerEndInfos
+FromMeshImpedance 0..1
0..*
+FromMeshImpedance
0..* +FromTransformerEndInfo 0..1 0..1
+TransformerEndInfo
+TransformerEndInfo
+StarImpedance 0..1
+CoreAdmittance
IdentifiedObject
Wires::TransformerCoreAdmittance
+ b: Susceptance [0..1]
+ b0: Susceptance [0..1]
+ g: Conductance [0..1]
+ g0: Conductance [0..1]
+CoreAdmittance
0..1 0..1
IdentifiedObject
Wires::TransformerStarImpedance
+ r: Resistance [0..1]
+ r0: Resistance [0..1]
+ x: Reactance [0..1]
+ x0: Reactance [0..1]
+TransformerStarImpedance
0..1
«enumeration» Core::PhaseCode {root}
ABCN
ABC
ABN
ACN
BCN
AB
AC
BC
AN
BN
CN
A
B
C
N
s1N
s2N
s12N
s1
s2
s12
«enumeration» Wires::WindingConnection {root}
D
Y
Z
Yn
Zn
A
I
IEC 289/13

Figure 12 – DCIM transformer connectivity model

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

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

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

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

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

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

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

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

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

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

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

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

4.4.3.4.2 Datasheet model

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

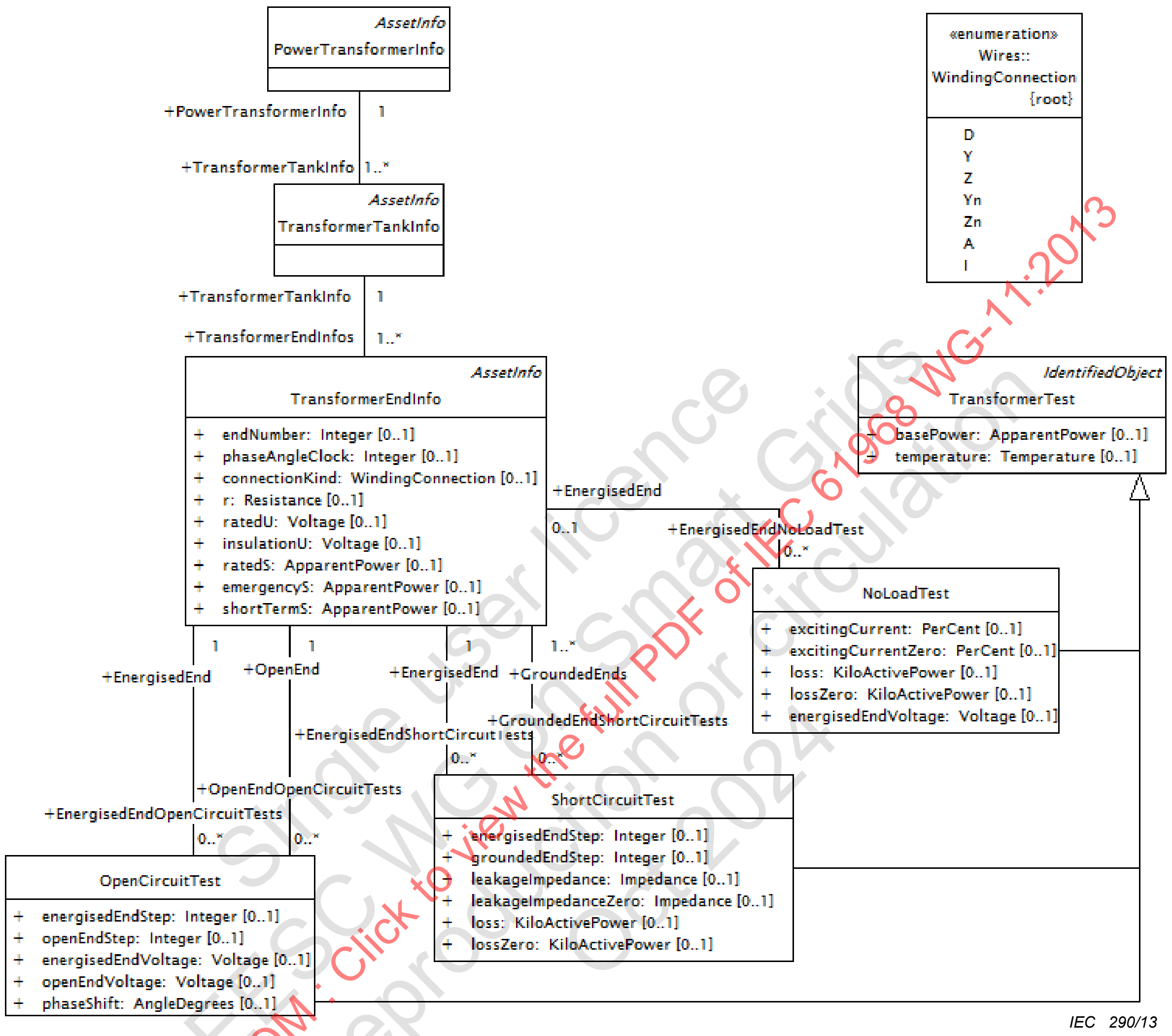

AssetInfo
PowerTransformerInfo
+PowerTransformerInfo 1
+TransformerTankInfo 1..*
AssetInfo
TransformerTankInfo
+TransformerTankInfo 1
+TransformerEndInfos 1..*
AssetInfo
TransformerEndInfo
+ endNumber: Integer [0..1]
+ phaseAngleClock: Integer [0..1]
+ connectionKind: WindingConnection [0..1]
+ r: Resistance [0..1]
+ ratedU: Voltage [0..1]
+ insulationU: Voltage [0..1]
+ ratedS: ApparentPower [0..1]
+ emergencyS: ApparentPower [0..1]
+ shortTermS: ApparentPower [0..1]
«enumeration» Wires:: WindingConnection {root}
D
Y
Z
Yn
Zn
A
I
IdentifiedObject
TransformerTest
+ basePower: ApparentPower [0..1]
+ temperature: Temperature [0..1]
+EnergisedEnd 0..1
+EnergisedEndNoLoadTest 0..*
NoLoadTest
+ excitingCurrent: PerCent [0..1]
+ excitingCurrentZero: PerCent [0..1]
+ loss: KiloActivePower [0..1]
+ lossZero: KiloActivePower [0..1]
+ energisedEndVoltage: Voltage [0..1]
1 +EnergisedEnd
1 +OpenEnd
1 +EnergisedEnd
1..* +GroundedEnds
+EnergisedEndShortCircuitTests 0..*
+GroundedEndShortCircuitTests 0..*
+OpenEndOpenCircuitTests 0..*
+EnergisedEndOpenCircuitTests 0..*
ShortCircuitTest
+ energisedEndStep: Integer [0..1]
+ groundedEndStep: Integer [0..1]
+ leakageImpedance: Impedance [0..1]
+ leakageImpedanceZero: Impedance [0..1]
+ loss: KiloActivePower [0..1]
+ lossZero: KiloActivePower [0..1]
OpenCircuitTest
+ energisedEndStep: Integer [0..1]
+ openEndStep: Integer [0..1]
+ energisedEndVoltage: Voltage [0..1]
+ openEndVoltage: Voltage [0..1]
+ phaseShift: AngleDegrees [0..1]
IEC 290/13

Figure 13 – DCIM transformer datasheet model

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

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

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

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

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

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

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

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

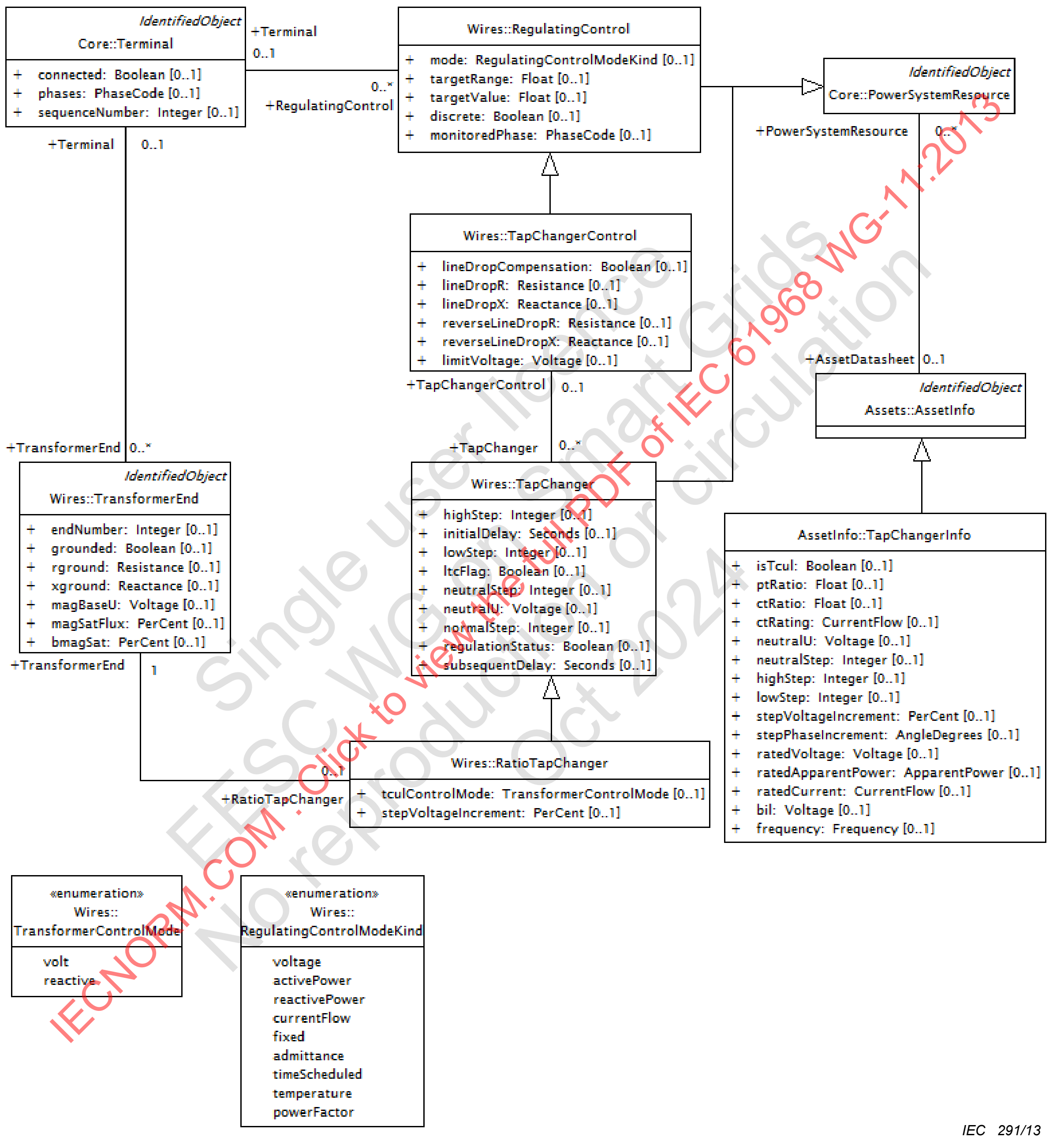
NoLoadTest's attributes **excitingCurrent**, **excitingCurrentZero**, **loss**, and **lossZero** are all measured on the **EnergisedEnd** winding and provide the basic data for a core admittance branch. Positive and zero sequence test data can be reported in the same instance of **NoLoadTest**. This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

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

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

4.4.3.4.3 Tap changer model

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



IdentifiedObject
Core::Terminal
+ connected: Boolean [0..1]
+ phases: PhaseCode [0..1]
+ sequenceNumber: Integer [0..1]
+Terminal
0..1
0..*
+RegulatingControl
Wires::RegulatingControl
+ mode: RegulatingControlModeKind [0..1]
+ targetRange: Float [0..1]
+ targetValue: Float [0..1]
+ discrete: Boolean [0..1]
+ monitoredPhase: PhaseCode [0..1]
IdentifiedObject
Core::PowerSystemResource
+PowerSystemResource
0..*
+Terminal
0..1
Wires::TapChangerControl
+ lineDropCompensation: Boolean [0..1]
+ lineDropR: Resistance [0..1]
+ lineDropX: Reactance [0..1]
+ reverseLineDropR: Resistance [0..1]
+ reverseLineDropX: Reactance [0..1]
+ limitVoltage: Voltage [0..1]
+AssetDatasheet
0..1
+TapChangerControl
0..1
IdentifiedObject
Assets::AssetInfo
+TransformerEnd
0..*
+TapChanger
0..*
IdentifiedObject
Wires::TransformerEnd
+ endNumber: Integer [0..1]
+ grounded: Boolean [0..1]
+ rground: Resistance [0..1]
+ xground: Reactance [0..1]
+ magBaseU: Voltage [0..1]
+ magSatFlux: PerCent [0..1]
+ bmagSat: PerCent [0..1]
Wires::TapChanger
+ highStep: Integer [0..1]
+ initialDelay: Seconds [0..1]
+ lowStep: Integer [0..1]
+ ltcFlag: Boolean [0..1]
+ neutralStep: Integer [0..1]
+ neutralU: Voltage [0..1]
+ normalStep: Integer [0..1]
+ regulationStatus: Boolean [0..1]
+ subsequentDelay: Seconds [0..1]
AssetInfo::TapChangerInfo
+ isTcul: Boolean [0..1]
+ ptRatio: Float [0..1]
+ ctRatio: Float [0..1]
+ ctRating: CurrentFlow [0..1]
+ neutralU: Voltage [0..1]
+ neutralStep: Integer [0..1]
+ highStep: Integer [0..1]
+ lowStep: Integer [0..1]
+ stepVoltageIncrement: PerCent [0..1]
+ stepPhaseIncrement: AngleDegrees [0..1]
+ ratedVoltage: Voltage [0..1]
+ ratedApparentPower: ApparentPower [0..1]
+ ratedCurrent: CurrentFlow [0..1]
+ bil: Voltage [0..1]
+ frequency: Frequency [0..1]
+TransformerEnd
1
0..1
+RatioTapChanger
Wires::RatioTapChanger
+ tculControlMode: TransformerControlMode [0..1]
+ stepVoltageIncrement: PerCent [0..1]
«enumeration»
Wires::
TransformerControlMode
volt
reactive
«enumeration»
Wires::
RegulatingControlModeKind
voltage
activePower
reactivePower
currentFlow
fixed
admittance
timeScheduled
temperature
powerFactor
IEC 291/13

Figure 14 – DCIM tap changer model

A three-phase line voltage regulator usually has three independent regulators to help correct voltage unbalance. The regulators are usually connected in wye. The model starts with a **PowerTransformer** containing three **TransformerTanks**, and a total of six **TransformerTankEnds**. There will be three instances of **RatioTapChanger**, each associated to a different **TransformerTankEnd**. The **RegulatingControl.monitoredPhase** attribute should be included among the **Terminal.phases** associated with the **PowerTransformer**. The tap positions, and sometimes the other attributes, will not be the same in each phase of the regulator. An open-delta regulator is also fairly common; this consists of two single-phase regulators connected line-to-line in a bank, with partial capability to correct voltage unbalance.

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

4.4.3.4.4 Example distribution transformer

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

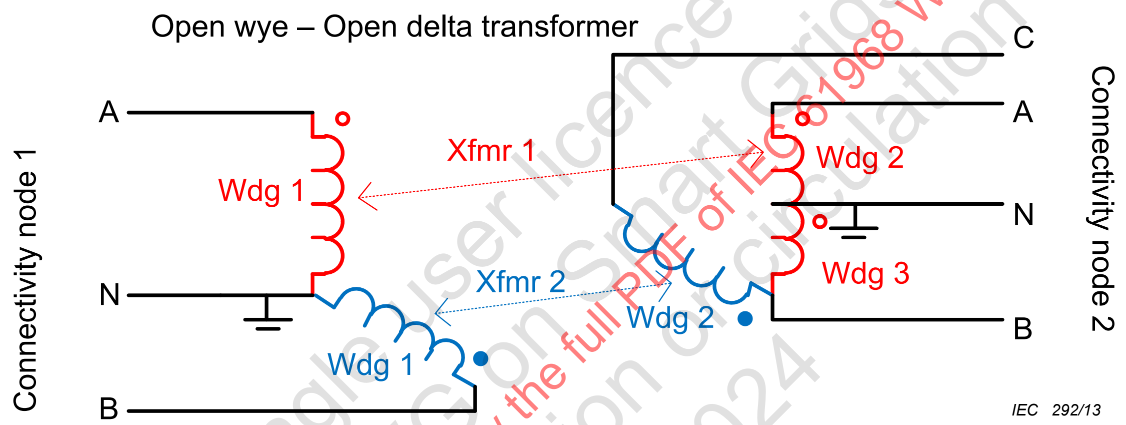


Figure 15 – Example of a distribution transformer that can be modelled with DCIM

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

Table 1 – Open wye/open delta transformer bank connections

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

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

4.4.3.4.5 Example autotransformer

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

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

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

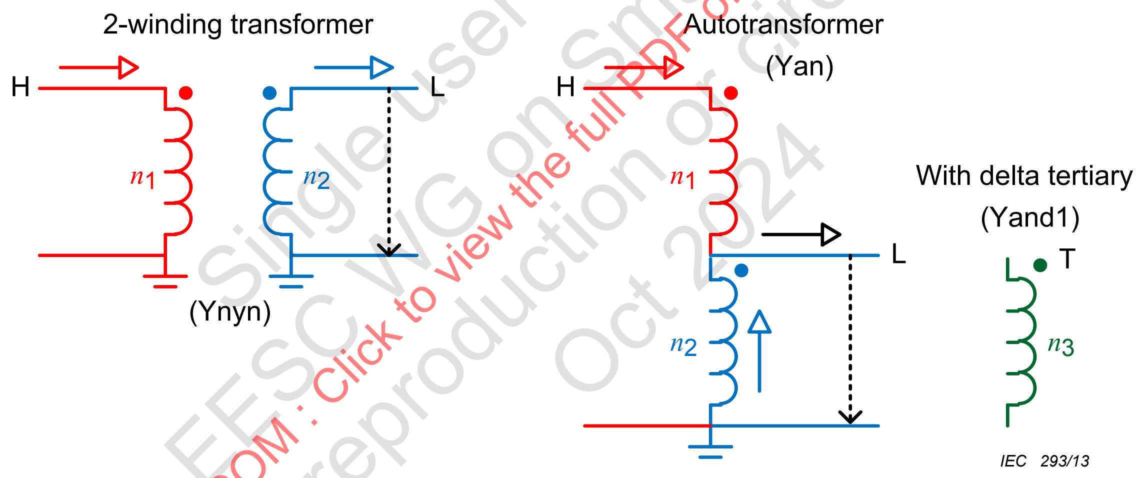


Figure 16 – Example of a two-winding transformer connected as an autotransformer

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

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

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

$$N = 1 + n_1 / n_2$$

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

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

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

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

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

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

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

4.4.3.5 Auxiliary equipment

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

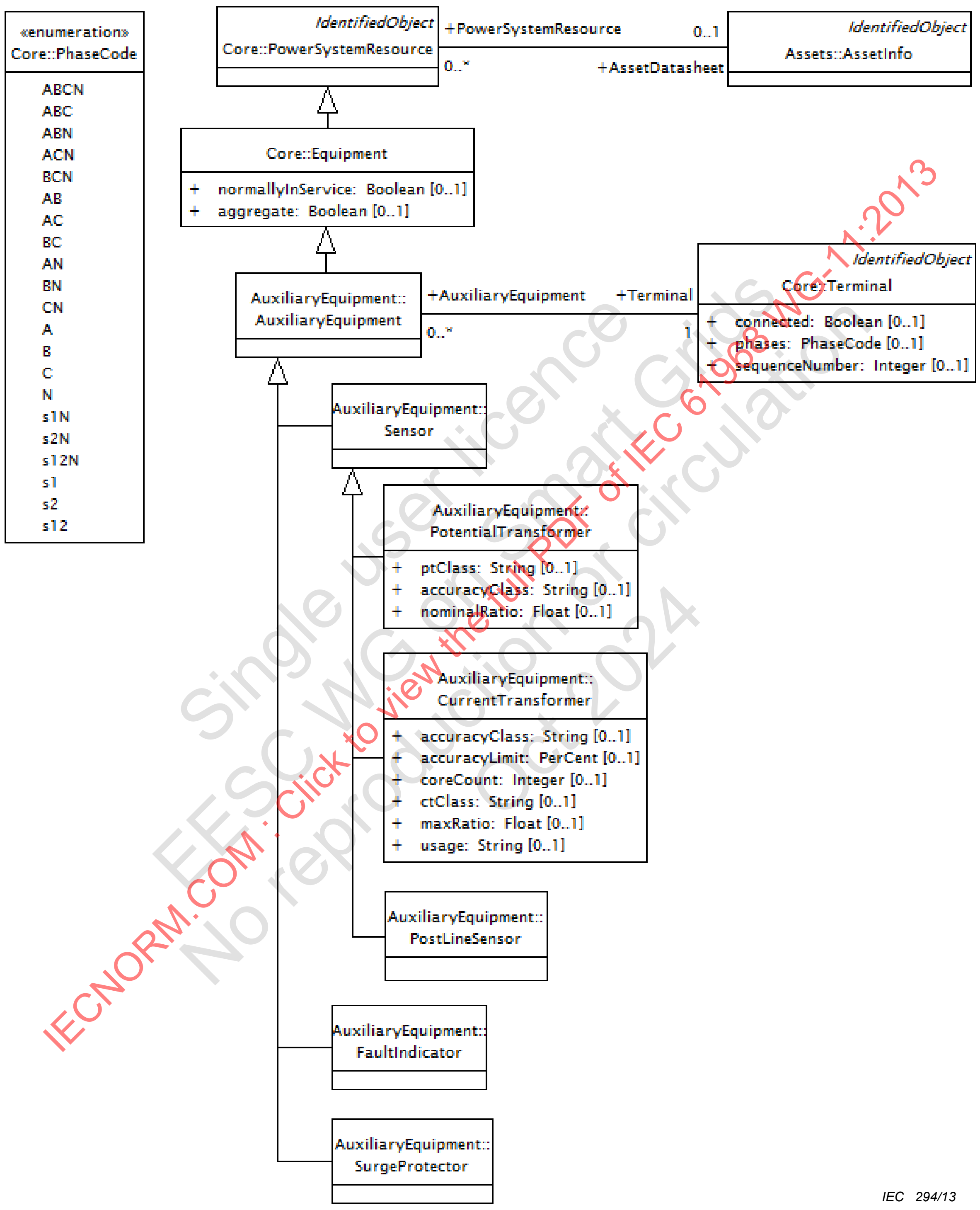


Figure 17 – DCIM auxiliary equipment

An important addition is the association of any **AuxiliaryEquipment** with a **Terminal**, which allows for positioning of the auxiliary devices in the electrical network connectivity model. For instance, it is now possible to explicitly model instrument transformers, position them on single line diagrams along with the usual **ConductingEquipment** and deduce the **Measurements** that they actually produce. This kind of data exchange would be typical when importing substation models defined with other data models than CIM.

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

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

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

4.4.4 Customers model

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

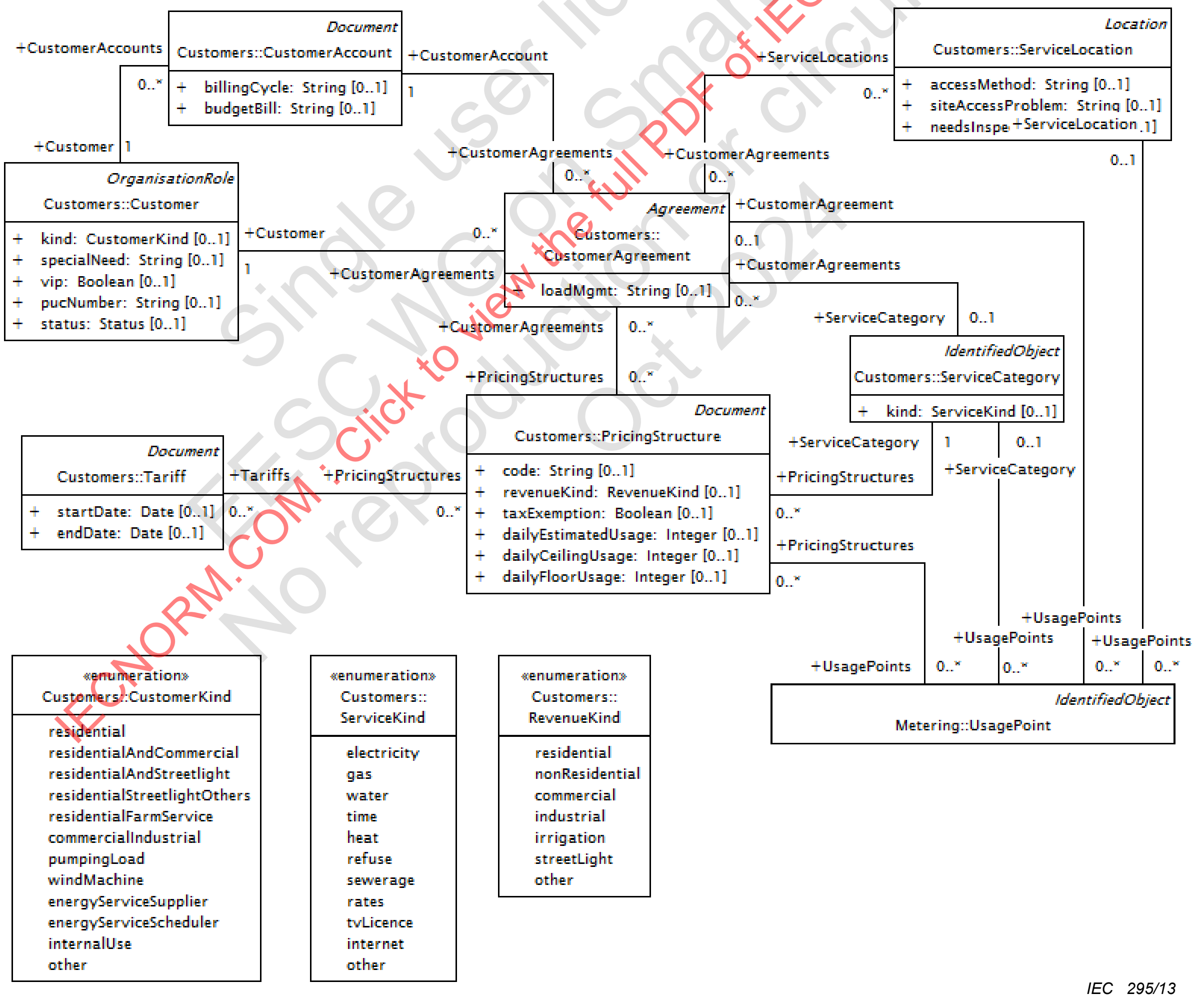


Figure 18 – DCIM customers model

A **Customer** is an organisation receiving services from one or more **ServiceSuppliers**. A **Customer** may have multiple **CustomerAccounts** and/or **CustomerAgreements**.

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

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

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

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

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

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

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

4.4.5 Metering model

4.4.5.1 General

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

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

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

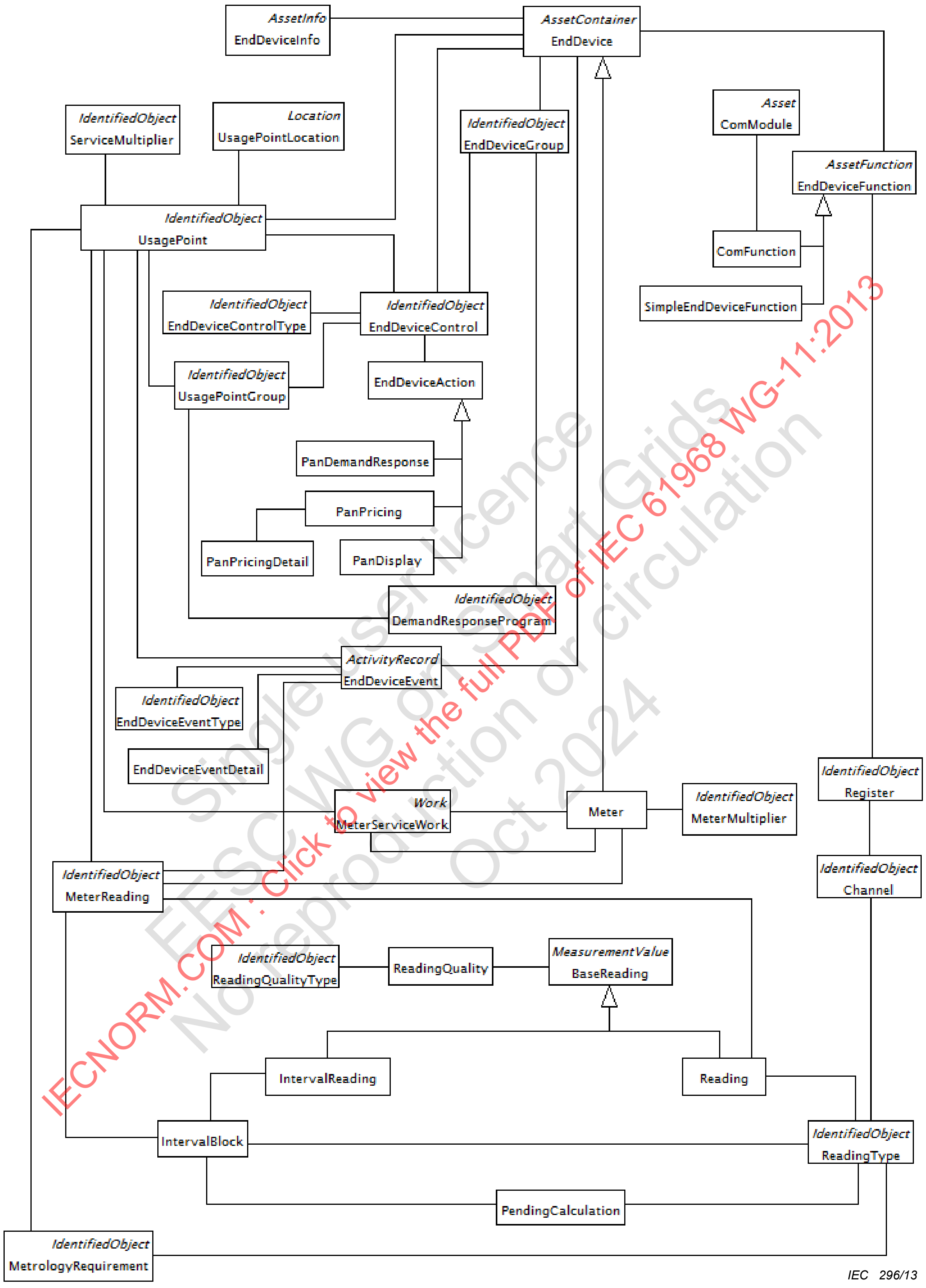


Figure 19 – DCIM metering model

These classes support typical metering domain entities, as well as concepts required for operational data exchanges. 4.4.5.2 to 4.4.5.7 describe related subsets in more detail. 4.4.5.2 through 4.4.5.4 focus mainly on the domain entities that typically need to be configured in order to enable operational data exchanges. These types of operational data exchanges are described in greater detail within 4.4.5.5 through 4.4.5.7.

4.4.5.2 Usage points

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

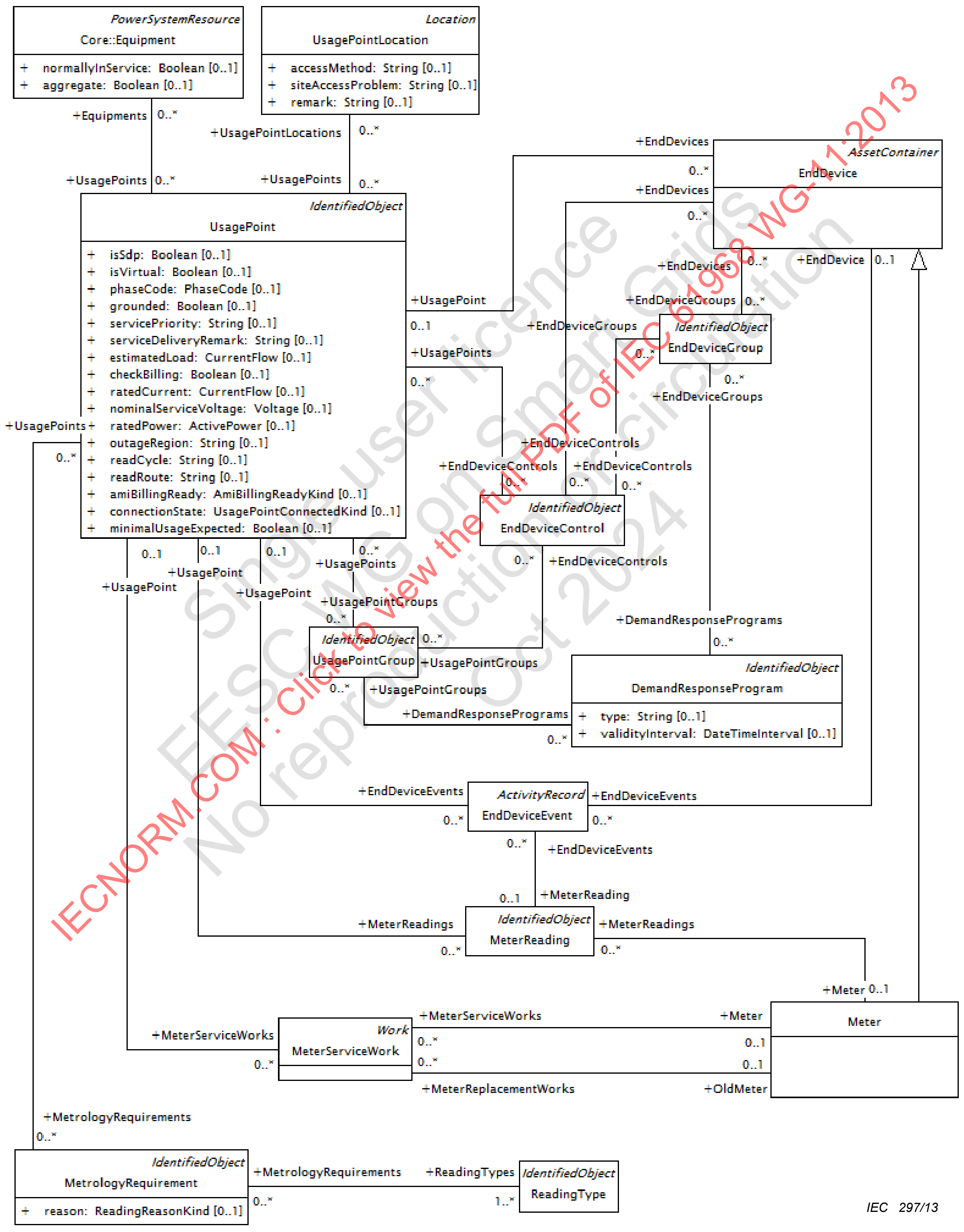


Figure 20 – DCIM usage point model

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

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

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

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

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

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

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

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

4.4.5.3 End devices

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

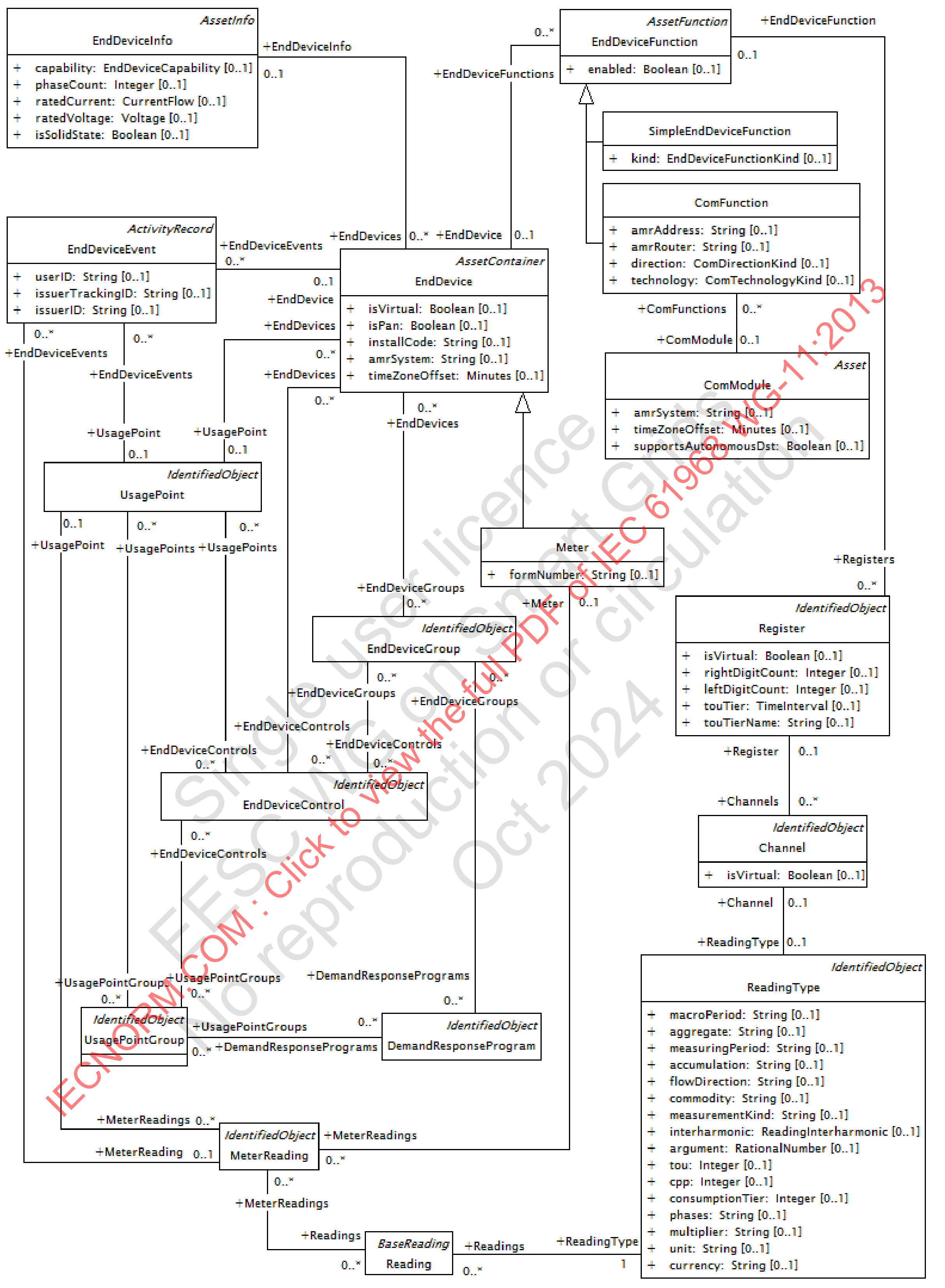


Figure 21 – DCIM end device model

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

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

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

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

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

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

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

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

4.4.5.4 Configuration events

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

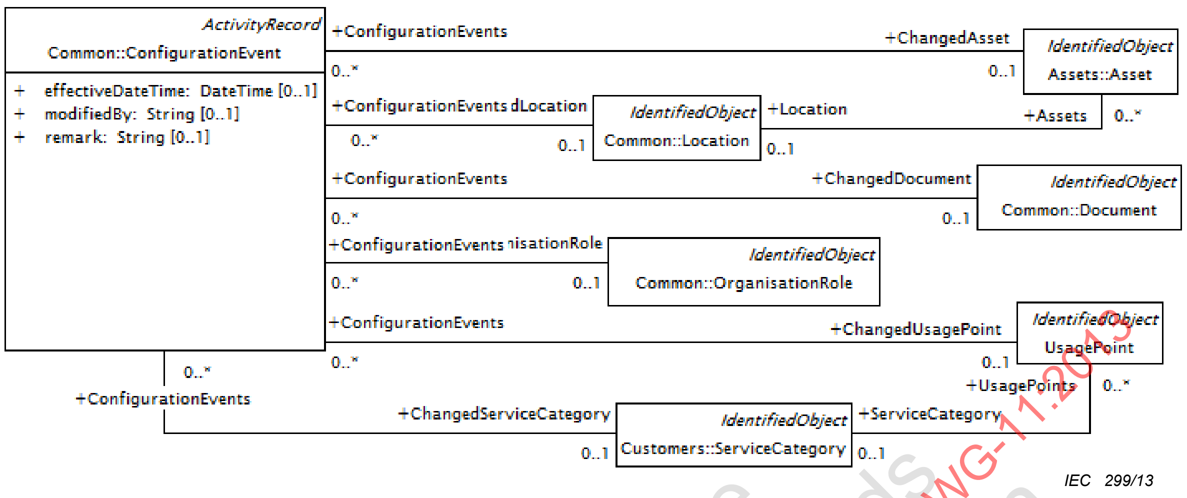


Figure 22 – Configuration events for metering

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

4.4.5.5 Meter readings

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

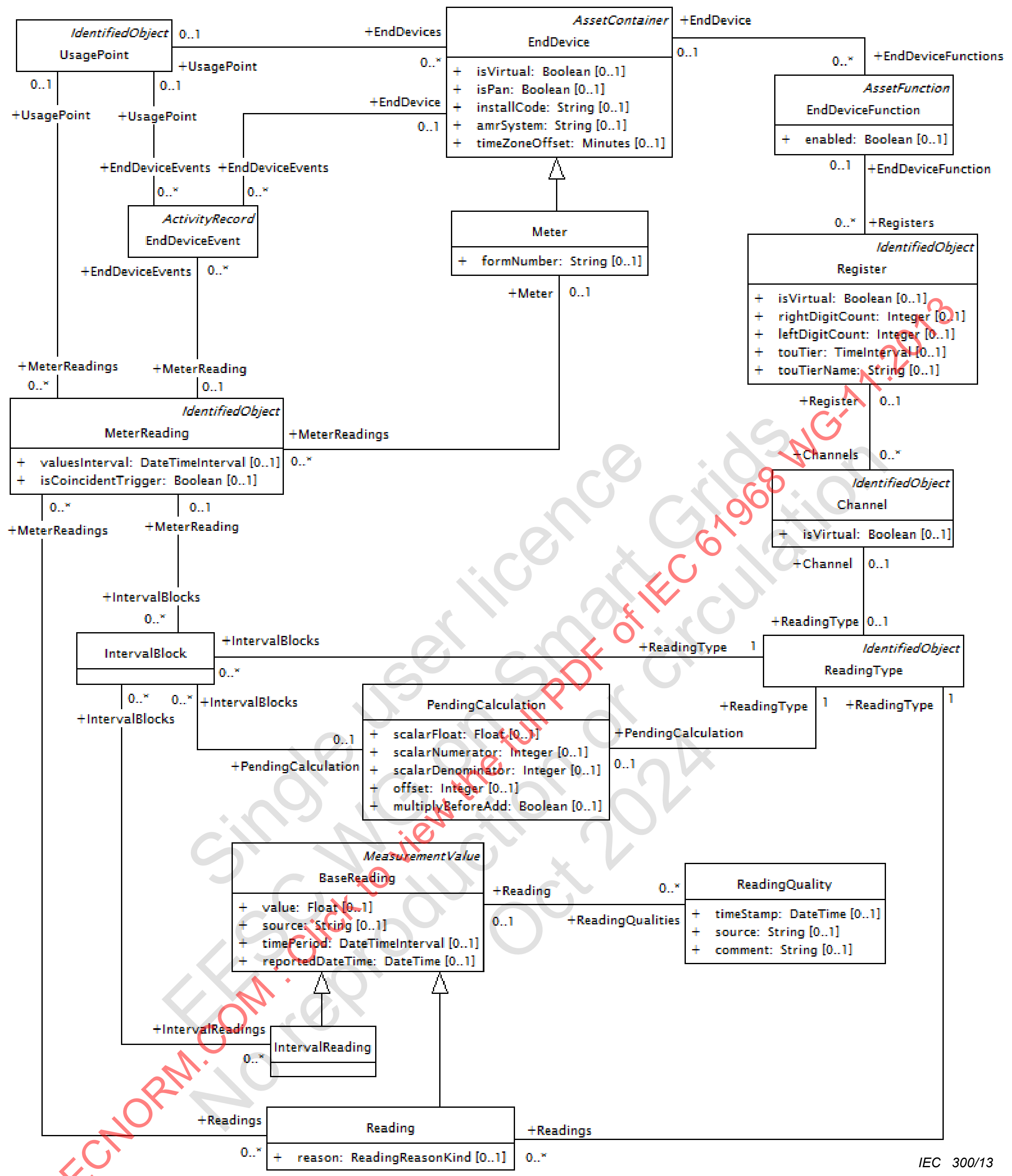


Figure 23 – DCIM meter readings model

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

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

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

4.4.5.6 End device controls and events

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

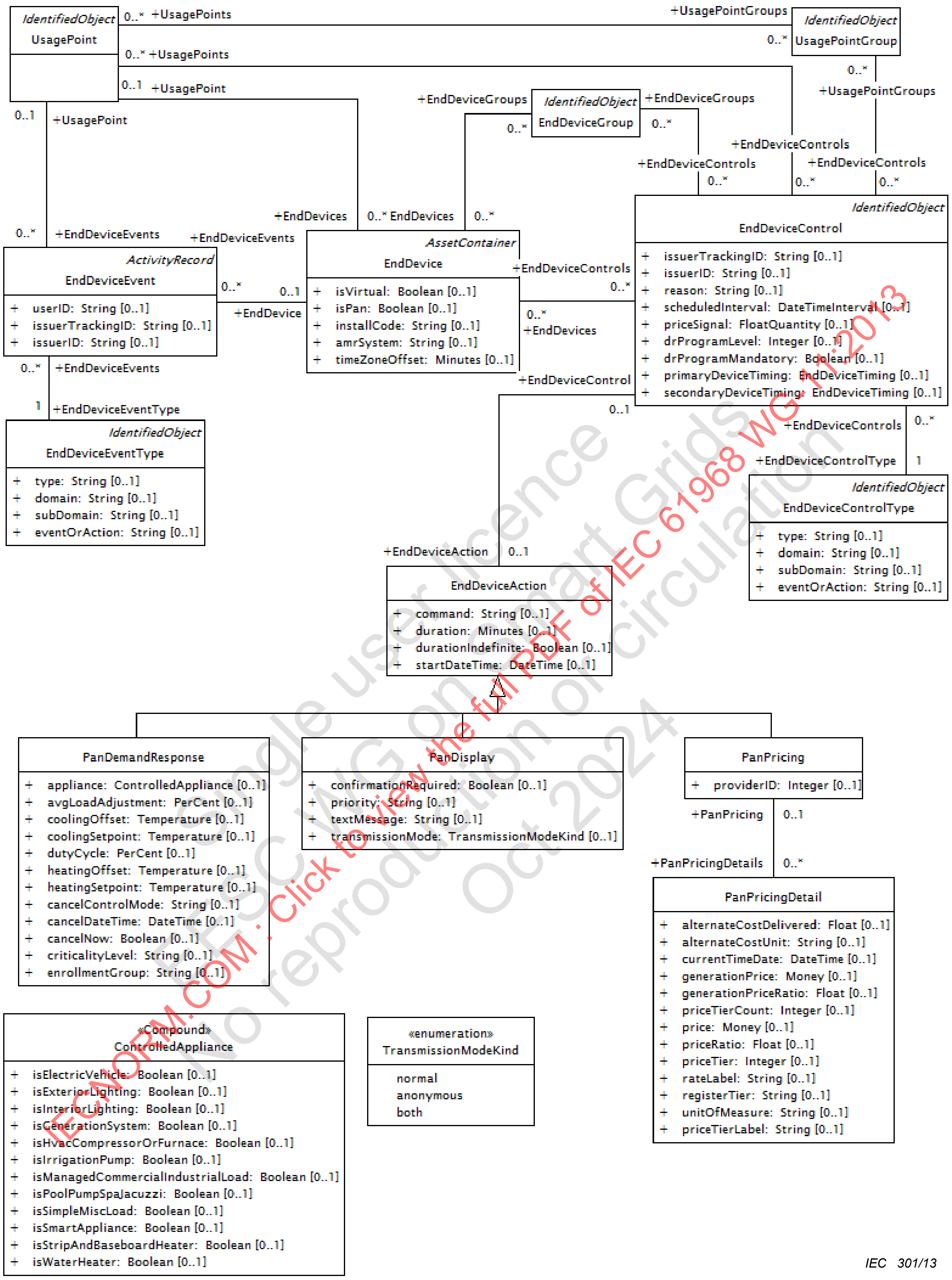

IdentifiedObject
UsagePoint
0..* +UsagePoints
+UsagePointGroups
IdentifiedObject
UsagePointGroup
0..* +UsagePoints
0..*
0..1 +UsagePoint
0..1 +UsagePoint
0..*
+UsagePointGroups
+EndDeviceGroups
IdentifiedObject
EndDeviceGroup
+EndDeviceGroups
0..*
0..*
+EndDeviceControls
+EndDeviceControls
+EndDeviceControls
0..*
0..*
0..*
+EndDevices
0..* EndDevices
0..*
0..* +EndDeviceEvents
+EndDeviceEvents
ActivityRecord
EndDeviceEvent
+ userID: String [0..1]
+ issuerTrackingID: String [0..1]
+ issuerID: String [0..1]
0..*
0..1
+EndDevice
AssetContainer
EndDevice
+ isVirtual: Boolean [0..1]
+ isPan: Boolean [0..1]
+ installCode: String [0..1]
+ amrSystem: String [0..1]
+ timeZoneOffset: Minutes [0..1]
+EndDeviceControls
0..*
0..*
+EndDevices
IdentifiedObject
EndDeviceControl
+ issuerTrackingID: String [0..1]
+ issuerID: String [0..1]
+ reason: String [0..1]
+ scheduledInterval: DateTimeInterval [0..1]
+ priceSignal: FloatQuantity [0..1]
+ drProgramLevel: Integer [0..1]
+ drProgramMandatory: Boolean [0..1]
+ primaryDeviceTiming: EndDeviceTiming [0..1]
+ secondaryDeviceTiming: EndDeviceTiming [0..1]
0..* +EndDeviceEvents
1 +EndDeviceEventType
IdentifiedObject
EndDeviceEventType
+ type: String [0..1]
+ domain: String [0..1]
+ subDomain: String [0..1]
+ eventOrAction: String [0..1]
+EndDeviceControl
0..1
+EndDeviceControls
0..*
+EndDeviceControlType
1
IdentifiedObject
EndDeviceControlType
+ type: String [0..1]
+ domain: String [0..1]
+ subDomain: String [0..1]
+ eventOrAction: String [0..1]
+EndDeviceAction
0..1
EndDeviceAction
+ command: String [0..1]
+ duration: Minutes [0..1]
+ durationIndefinite: Boolean [0..1]
+ startDateTime: DateTime [0..1]
PanDemandResponse
+ appliance: ControlledAppliance [0..1]
+ avgLoadAdjustment: PerCent [0..1]
+ coolingOffset: Temperature [0..1]
+ coolingSetpoint: Temperature [0..1]
+ dutyCycle: PerCent [0..1]
+ heatingOffset: Temperature [0..1]
+ heatingSetpoint: Temperature [0..1]
+ cancelControlMode: String [0..1]
+ cancelDateTime: DateTime [0..1]
+ cancelNow: Boolean [0..1]
+ criticalityLevel: String [0..1]
+ enrollmentGroup: String [0..1]
PanDisplay
+ confirmationRequired: Boolean [0..1]
+ priority: String [0..1]
+ textMessage: String [0..1]
+ transmissionMode: TransmissionModeKind [0..1]
PanPricing
+ providerID: Integer [0..1]
+PanPricing
0..1
+PanPricingDetails
0..*
PanPricingDetail
+ alternateCostDelivered: Float [0..1]
+ alternateCostUnit: String [0..1]
+ currentTimeDate: DateTime [0..1]
+ generationPrice: Money [0..1]
+ generationPriceRatio: Float [0..1]
+ priceTierCount: Integer [0..1]
+ price: Money [0..1]
+ priceRatio: Float [0..1]
+ priceTier: Integer [0..1]
+ rateLabel: String [0..1]
+ registerTier: String [0..1]
+ unitOfMeasure: String [0..1]
+ priceTierLabel: String [0..1]
«Compound»
ControlledAppliance
+ isElectricVehicle: Boolean [0..1]
+ isExteriorLighting: Boolean [0..1]
+ isInteriorLighting: Boolean [0..1]
+ isGenerationSystem: Boolean [0..1]
+ isHvacCompressorOrFurnace: Boolean [0..1]
+ isIrrigationPump: Boolean [0..1]
+ isManagedCommercialIndustrialLoad: Boolean [0..1]
+ isPoolPumpSpaJacuzzi: Boolean [0..1]
+ isSimpleMiscLoad: Boolean [0..1]
+ isSmartAppliance: Boolean [0..1]
+ isStripAndBaseboardHeater: Boolean [0..1]
+ isWaterHeater: Boolean [0..1]
«enumeration»
TransmissionModeKind
normal
anonymous
both
IEC 301/13

Figure 24 – DCIM end device controls and events model

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

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

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

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

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

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

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

4.4.5.7 Meter service requests

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

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

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

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

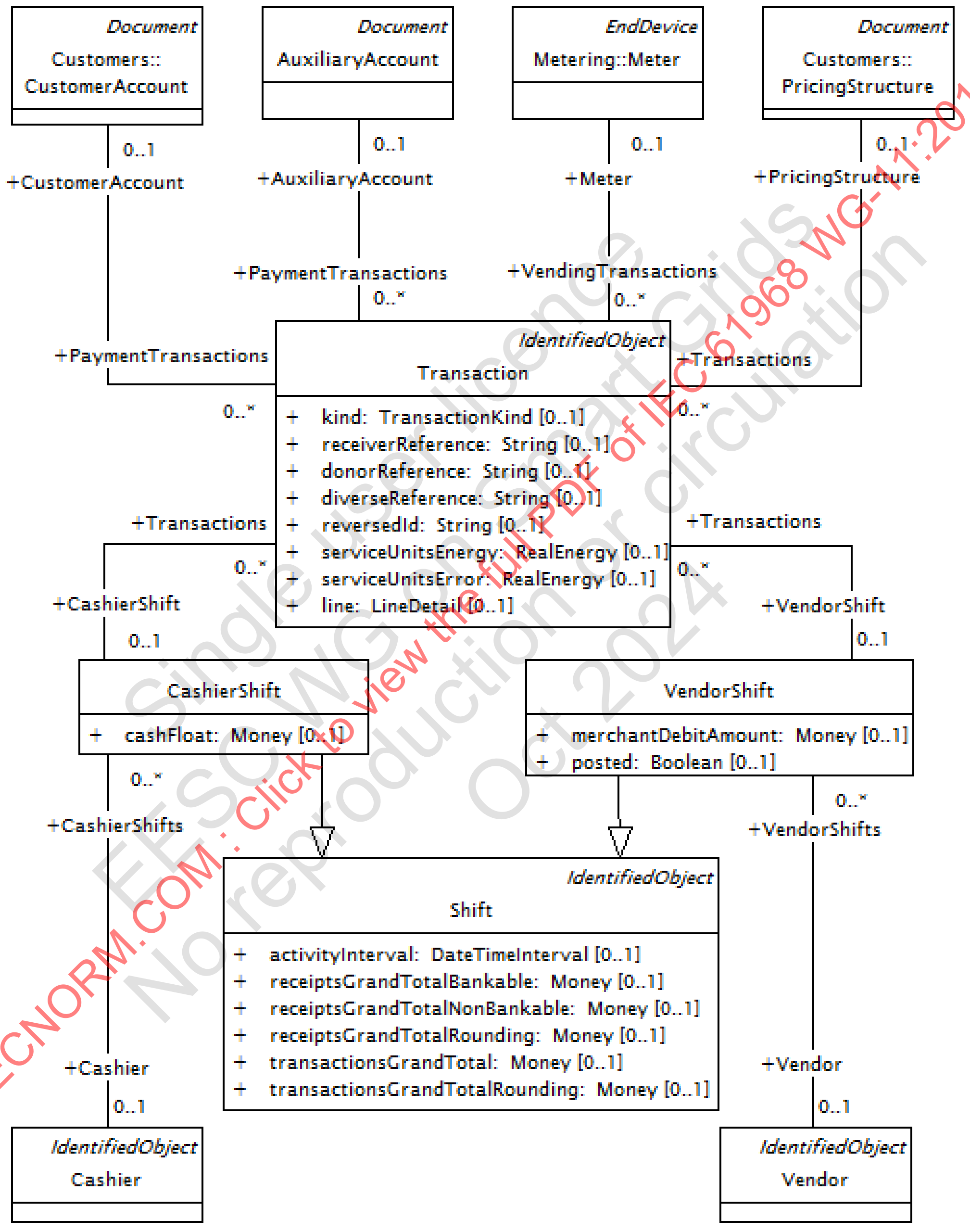
4.4.6 Payment metering

4.4.6.1 General

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

4.4.6.2 Transacting

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



Document
Customers::CustomerAccount
Document
AuxiliaryAccount
EndDevice
Metering::Meter
Document
Customers::PricingStructure
0..1
+CustomerAccount
+AuxiliaryAccount
+Meter
+PricingStructure
+PaymentTransactions
+VendingTransactions
0..*
IdentifiedObject
Transaction
+ kind: TransactionKind [0..1]
+ receiverReference: String [0..1]
+ donorReference: String [0..1]
+ diverseReference: String [0..1]
+ reversedId: String [0..1]
+ serviceUnitsEnergy: RealEnergy [0..1]
+ serviceUnitsError: RealEnergy [0..1]
+ line: LineDetail [0..1]
+Transactions
+CashierShift
+VendorShift
CashierShift
+ cashFloat: Money [0..1]
VendorShift
+ merchantDebitAmount: Money [0..1]
+ posted: Boolean [0..1]
+CashierShifts
+VendorShifts
IdentifiedObject
Shift
+ activityInterval: DateTimeInterval [0..1]
+ receiptsGrandTotalBankable: Money [0..1]
+ receiptsGrandTotalNonBankable: Money [0..1]
+ receiptsGrandTotalRounding: Money [0..1]
+ transactionsGrandTotal: Money [0..1]
+ transactionsGrandTotalRounding: Money [0..1]
+Cashier
+Vendor
IdentifiedObject
Cashier
IdentifiedObject
Vendor
IEC 302/13

Figure 25 – DCIM transacting model

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

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

4.4.6.3 Receipting

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

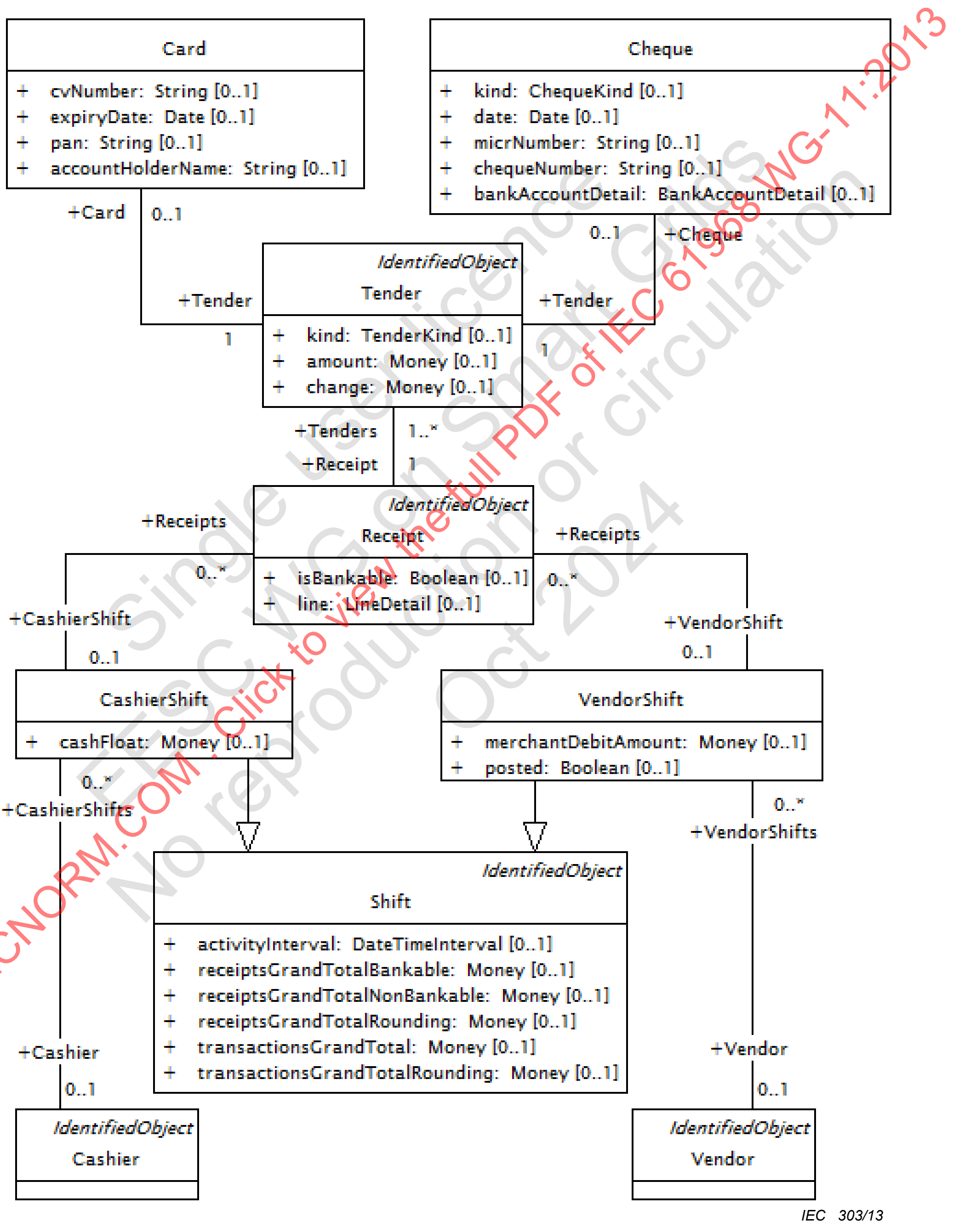

Card
+ cvNumber: String [0..1]
+ expiryDate: Date [0..1]
+ pan: String [0..1]
+ accountHolderName: String [0..1]
Cheque
+ kind: ChequeKind [0..1]
+ date: Date [0..1]
+ micrNumber: String [0..1]
+ chequeNumber: String [0..1]
+ bankAccountDetail: BankAccountDetail [0..1]
+Card
0..1
+Tender
1
0..1
+Cheque
+Tender
1
IdentifiedObject
Tender
+ kind: TenderKind [0..1]
+ amount: Money [0..1]
+ change: Money [0..1]
+Tenders
1..*
+Receipt
1
IdentifiedObject
Receipt
+ isBankable: Boolean [0..1]
+ line: LineDetail [0..1]
+Receipts
0..*
+Receipts
0..*
+CashierShift
0..1
+VendorShift
0..1
CashierShift
+ cashFloat: Money [0..1]
VendorShift
+ merchantDebitAmount: Money [0..1]
+ posted: Boolean [0..1]
0..*
+CashierShifts
0..*
+VendorShifts
IdentifiedObject
Shift
+ activityInterval: DateTimeInterval [0..1]
+ receiptsGrandTotalBankable: Money [0..1]
+ receiptsGrandTotalNonBankable: Money [0..1]
+ receiptsGrandTotalRounding: Money [0..1]
+ transactionsGrandTotal: Money [0..1]
+ transactionsGrandTotalRounding: Money [0..1]
+Cashier
0..1
+Vendor
0..1
IdentifiedObject
Cashier
IdentifiedObject
Vendor
IEC 303/13

Figure 26 – DCIM receipting model

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

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

4.4.6.4 Auxiliary payments

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

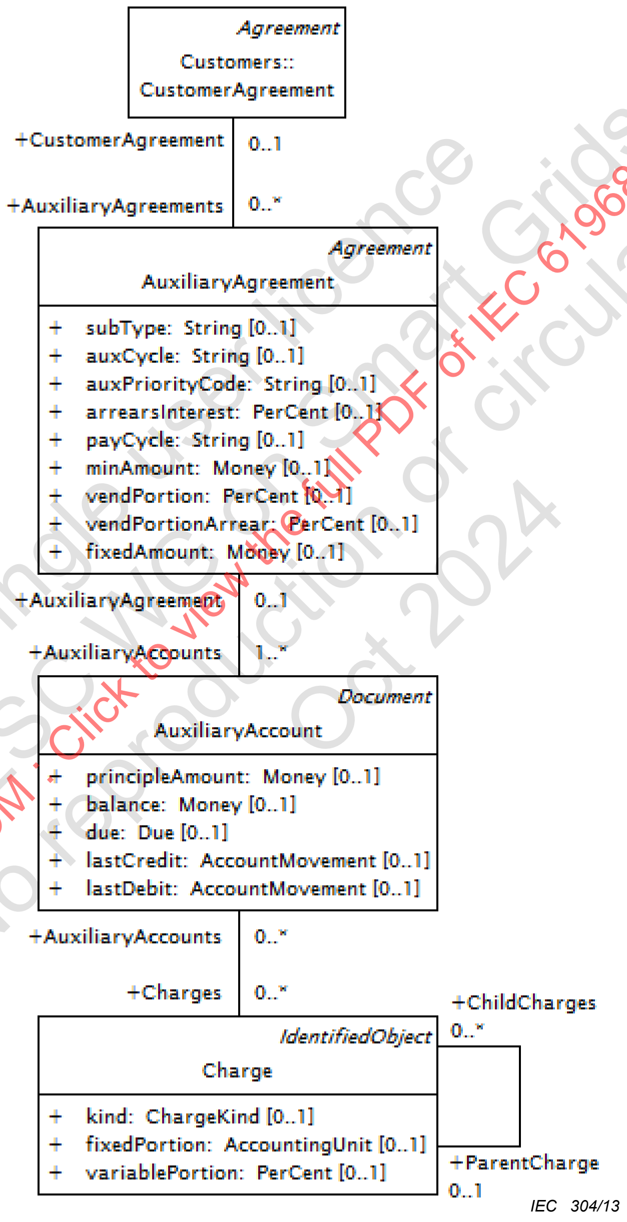

Agreement
Customers::
CustomerAgreement
+CustomerAgreement
0..1
+AuxiliaryAgreements
0..*
Agreement
AuxiliaryAgreement
+ subType: String [0..1]
+ auxCycle: String [0..1]
+ auxPriorityCode: String [0..1]
+ arrearsInterest: PerCent [0..1]
+ payCycle: String [0..1]
+ minAmount: Money [0..1]
+ vendPortion: PerCent [0..1]
+ vendPortionArrear: PerCent [0..1]
+ fixedAmount: Money [0..1]
+AuxiliaryAgreement
0..1
+AuxiliaryAccounts
1..*
Document
AuxiliaryAccount
+ principleAmount: Money [0..1]
+ balance: Money [0..1]
+ due: Due [0..1]
+ lastCredit: AccountMovement [0..1]
+ lastDebit: AccountMovement [0..1]
+AuxiliaryAccounts
0..*
+Charges
0..*
+ChildCharges
0..*
IdentifiedObject
Charge
+ kind: ChargeKind [0..1]
+ fixedPortion: AccountingUnit [0..1]
+ variablePortion: PerCent [0..1]
+ParentCharge
0..1
IEC 304/13

Figure 27 – DCIM auxiliary agreement model

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

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

4.4.6.5 Pricing and tariff structures

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

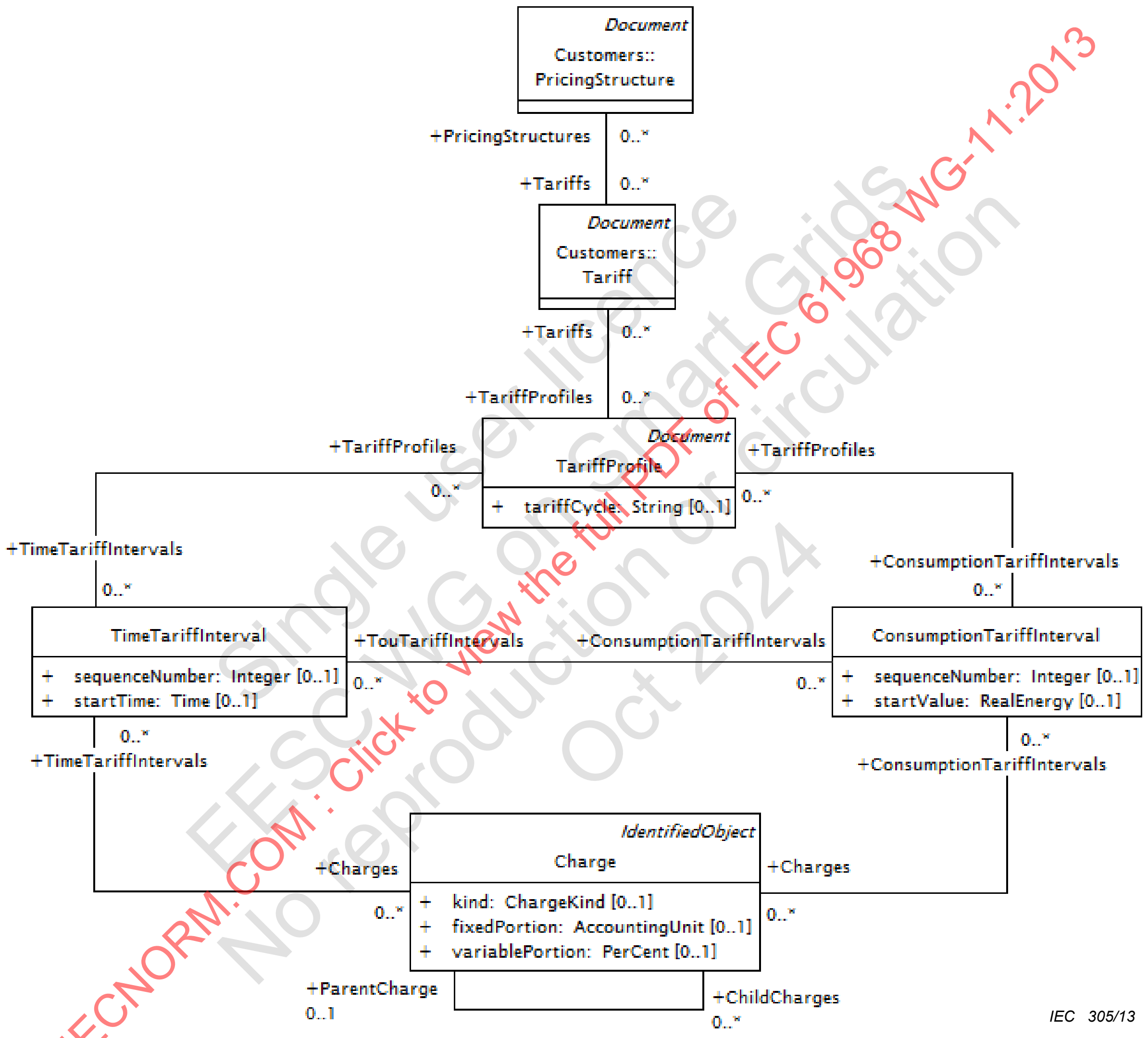


Figure 28 – DCIM pricing structure model

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

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

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

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

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

4.5 Other

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

5 Detailed model

5.1 Overview

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

5.2 Context

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

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

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

Table 2 – Attribute documentation

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

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

Table 3 – Association ends documentation

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

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

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

Table 4 – Enums documentation

literal	description
steel	
lead	
lock	
other	

6 Top package IEC61968

6.1 General

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

Figure 29 shows package diagram IEC61968Dependencies.

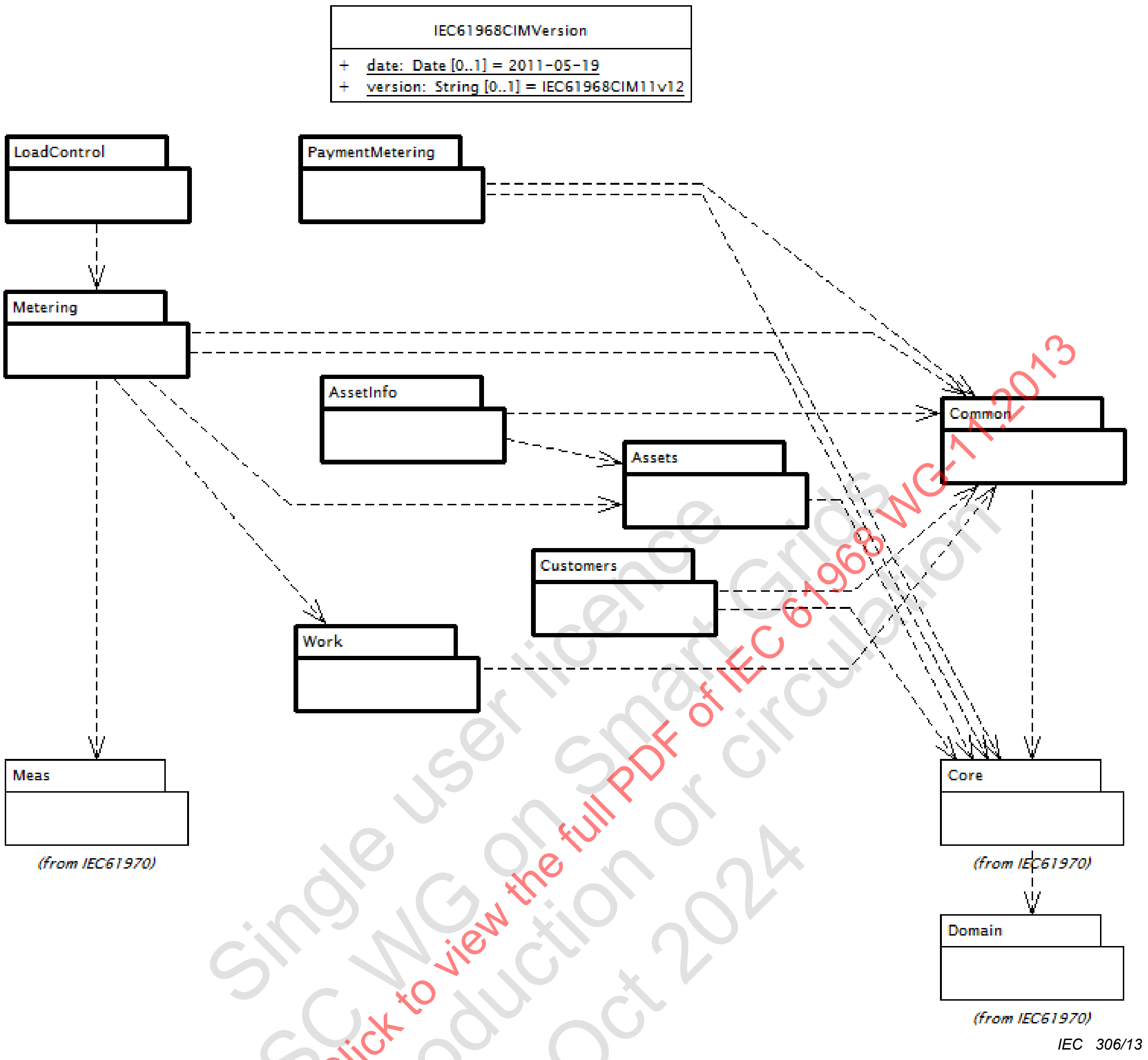


Figure 29 – Package diagram IEC61968::IEC61968Dependencies

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

Packages in bold are contained and documented in Clause 6.

6.2 IEC61968CIMVersion root class

IEC 61968 version number assigned to this UML model.

Table 5 shows all attributes of IEC61968CIMVersion.

Table 5 – Attributes of IEC61968::IEC61968CIMVersion

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

6.3 Package Common

6.3.1 General

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

Figure 30 shows class diagram CommonInheritance.

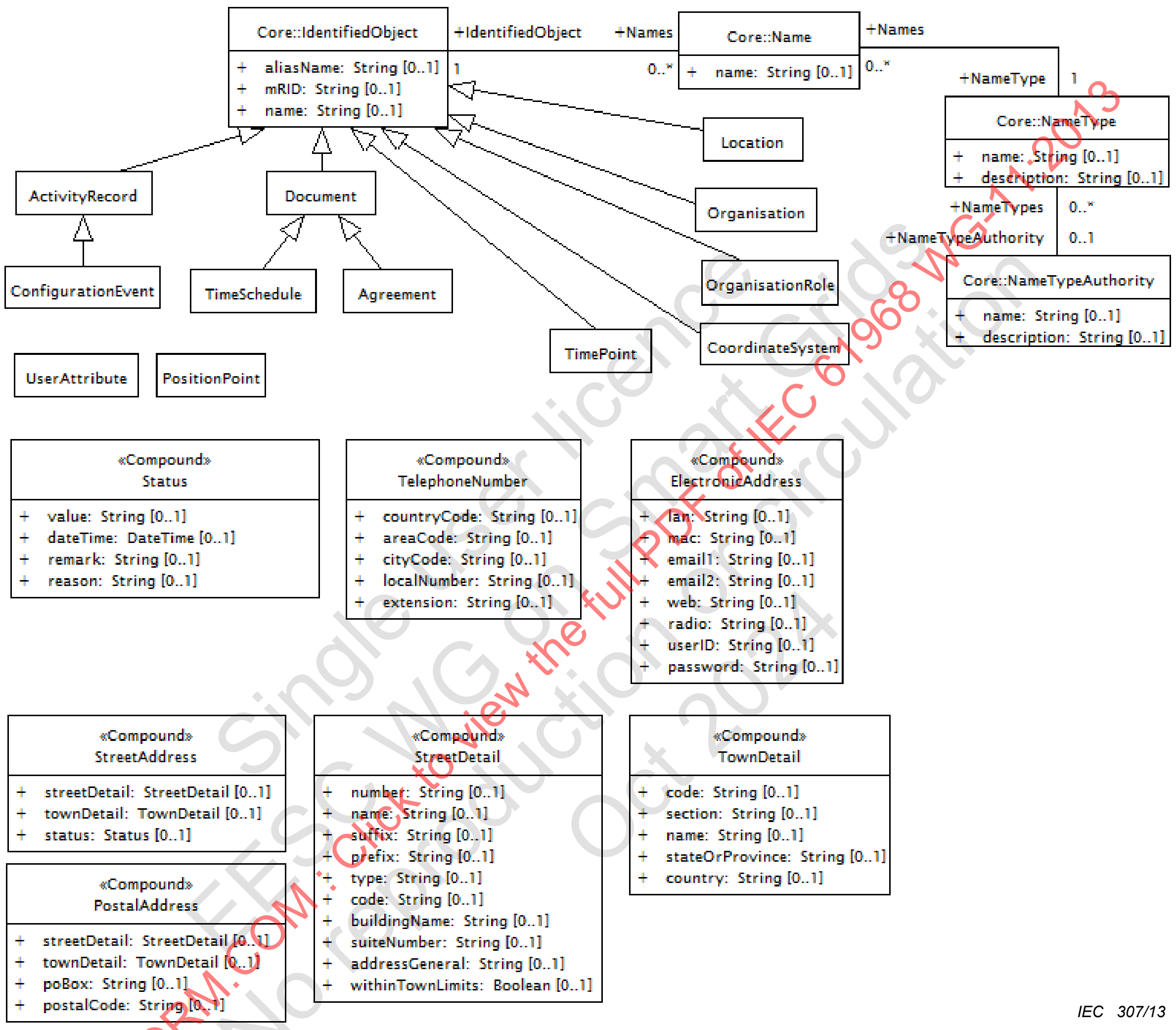


Figure 30 – Class diagram Common::CommonInheritance

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

Figure 31 shows class diagram CommonOverview.

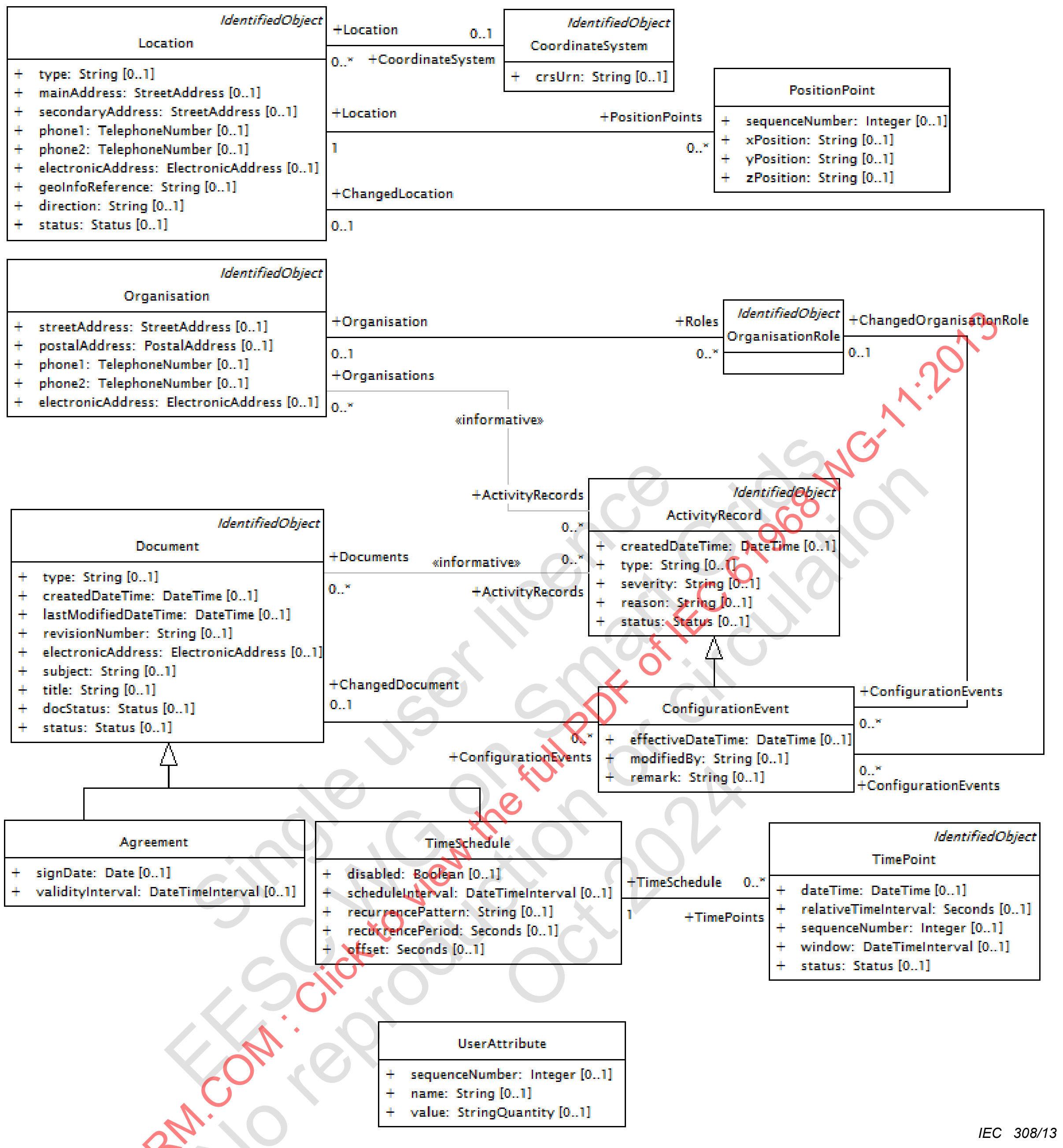

IdentifiedObject
Location
+ type: String [0..1]
+ mainAddress: StreetAddress [0..1]
+ secondaryAddress: StreetAddress [0..1]
+ phone1: TelephoneNumber [0..1]
+ phone2: TelephoneNumber [0..1]
+ electronicAddress: ElectronicAddress [0..1]
+ geoInfoReference: String [0..1]
+ direction: String [0..1]
+ status: Status [0..1]
IdentifiedObject
CoordinateSystem
+ crsUrn: String [0..1]
PositionPoint
+ sequenceNumber: Integer [0..1]
+ xPosition: String [0..1]
+ yPosition: String [0..1]
+ zPosition: String [0..1]
IdentifiedObject
Organisation
+ streetAddress: StreetAddress [0..1]
+ postalAddress: PostalAddress [0..1]
+ phone1: TelephoneNumber [0..1]
+ phone2: TelephoneNumber [0..1]
+ electronicAddress: ElectronicAddress [0..1]
IdentifiedObject
OrganisationRole
IdentifiedObject
ActivityRecord
+ createdDateTime: DateTime [0..1]
+ type: String [0..1]
+ severity: String [0..1]
+ reason: String [0..1]
+ status: Status [0..1]
IdentifiedObject
Document
+ type: String [0..1]
+ createdDateTime: DateTime [0..1]
+ lastModifiedDateTime: DateTime [0..1]
+ revisionNumber: String [0..1]
+ electronicAddress: ElectronicAddress [0..1]
+ subject: String [0..1]
+ title: String [0..1]
+ docStatus: Status [0..1]
+ status: Status [0..1]
ConfigurationEvent
+ effectiveDateTime: DateTime [0..1]
+ modifiedBy: String [0..1]
+ remark: String [0..1]
Agreement
+ signDate: Date [0..1]
+ validityInterval: DateTimeInterval [0..1]
TimeSchedule
+ disabled: Boolean [0..1]
+ scheduleInterval: DateTimeInterval [0..1]
+ recurrencePattern: String [0..1]
+ recurrencePeriod: Seconds [0..1]
+ offset: Seconds [0..1]
IdentifiedObject
TimePoint
+ dateTime: DateTime [0..1]
+ relativeTimeInterval: Seconds [0..1]
+ sequenceNumber: Integer [0..1]
+ window: DateTimeInterval [0..1]
+ status: Status [0..1]
UserAttribute
+ sequenceNumber: Integer [0..1]
+ name: String [0..1]
+ value: StringQuantity [0..1]
+Location 0..1
0..* +CoordinateSystem
+Location
+PositionPoints
1
0..*
+ChangedLocation
0..1
+Organisation
+Roles
0..1
0..*
+ChangedOrganisationRole
0..1
+Organisations
0..*
«informative»
+ActivityRecords
0..*
+Documents
«informative»
0..*
0..*
+ActivityRecords
+ChangedDocument
0..1
+ConfigurationEvents
0..*
0..*
+ConfigurationEvents
0..*
+ConfigurationEvents
+TimeSchedule 0..*
1
+TimePoints
IEC 308/13

Figure 31 – Class diagram Common::CommonOverview

This diagram shows normative classes from this package.

6.3.2 Status compound

Current status information relevant to an entity.

Table 6 shows all attributes of Status.

Table 6 – Attributes of Common::Status

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

6.3.3 PostalAddress compound

General purpose postal address information.

Table 7 shows all attributes of PostalAddress.

Table 7 – Attributes of Common::PostalAddress

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

6.3.4 StreetAddress compound

General purpose street address information.

Table 8 shows all attributes of StreetAddress.

Table 8 – Attributes of Common::StreetAddress

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

6.3.5 StreetDetail compound

Street details, in the context of address.

Table 9 shows all attributes of StreetDetail.

Table 9 – Attributes of Common::StreetDetail

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

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

6.3.6 TownDetail compound

Town details, in the context of address.

Table 10 shows all attributes of TownDetail.

Table 10 – Attributes of Common::TownDetail

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

6.3.7 ElectronicAddress compound

Electronic address information.

Table 11 shows all attributes of ElectronicAddress.

Table 11 – Attributes of Common::ElectronicAddress

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

6.3.8 TelephoneNumber compound

Telephone number.

Table 12 shows all attributes of TelephoneNumber.

Table 12 – Attributes of Common::TelephoneNumber

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

6.3.9 ActivityRecord

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

Table 13 shows all attributes of ActivityRecord.

Table 13 – Attributes of Common::ActivityRecord

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

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

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

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

6.3.10 Agreement

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

Table 15 shows all attributes of Agreement.

Table 15 – Attributes of Common::Agreement

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

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

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

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

6.3.11 ConfigurationEvent

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

Table 17 shows all attributes of ConfigurationEvent.

Table 17 – Attributes of Common::ConfigurationEvent

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

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

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

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

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

6.3.12 CoordinateSystem

Coordinate reference system.

Table 19 shows all attributes of CoordinateSystem.

Table 19 – Attributes of Common::CoordinateSystem

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

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

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

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

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

6.3.13 Document

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

Table 21 shows all attributes of Document.

Table 21 – Attributes of Common::Document

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

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

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

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

6.3.14 Location

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

Table 23 shows all attributes of Location.

Table 23 – Attributes of Common::Location

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

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

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

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

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

6.3.15 Organisation

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

Table 25 shows all attributes of Organisation.

Table 25 – Attributes of Common::Organisation

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

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

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

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

6.3.16 OrganisationRole

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

Table 27 shows all attributes of OrganisationRole.

Table 27 – Attributes of Common::OrganisationRole

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

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

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

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

6.3.17 PositionPoint root class

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

Table 29 shows all attributes of PositionPoint.

Table 29 – Attributes of Common::PositionPoint

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

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

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

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

6.3.18 TimePoint

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

Table 31 shows all attributes of TimePoint.

Table 31 – Attributes of Common::TimePoint

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

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

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

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

6.3.19 TimeSchedule

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

Table 33 shows all attributes of TimeSchedule.

Table 33 – Attributes of Common::TimeSchedule

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

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

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

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

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

6.3.20 UserAttribute root class

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

Table 35 shows all attributes of UserAttribute.

Table 35 – Attributes of Common::UserAttribute

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

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

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

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

6.4 Package Assets

6.4.1 General

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

Figure 32 shows class diagram AssetsInheritance.

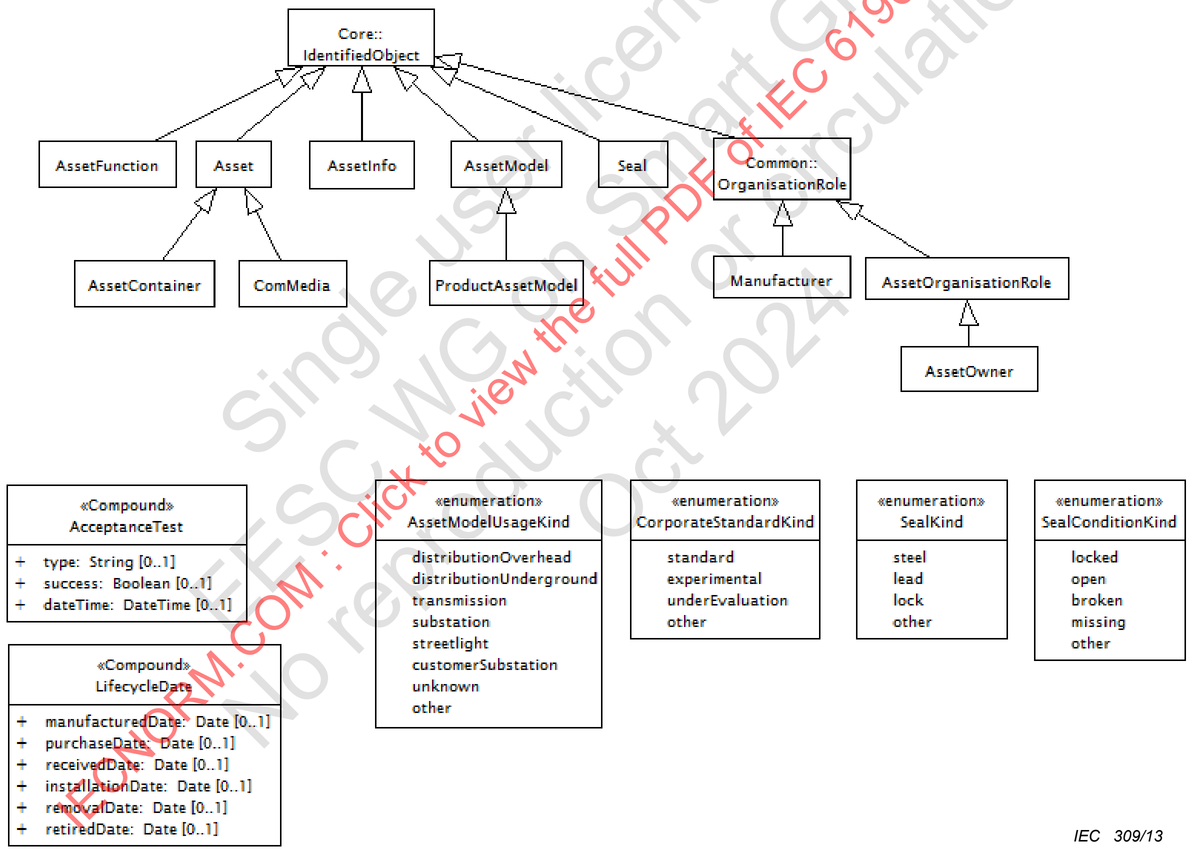
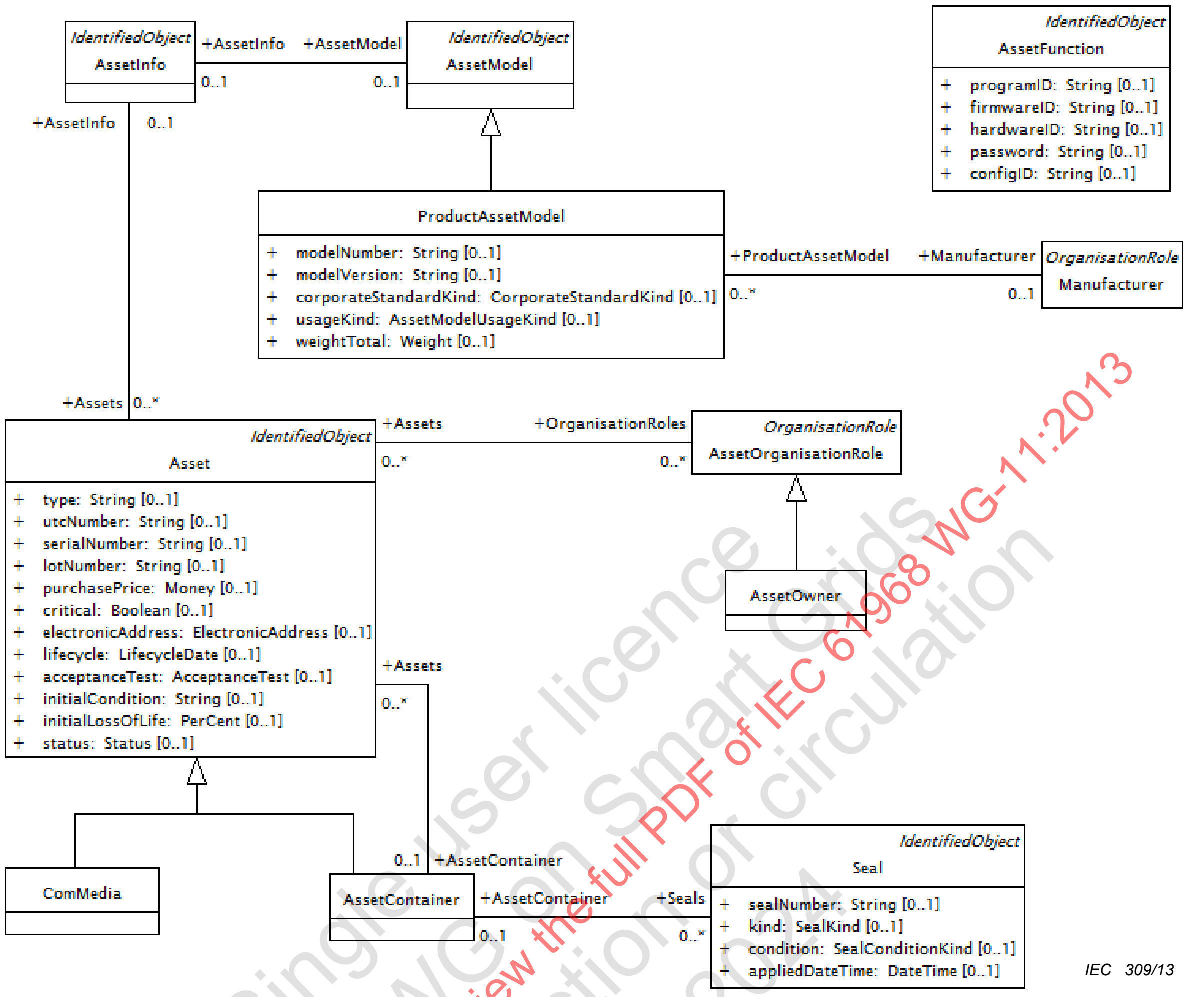


Figure 32 – Class diagram Assets::AssetsInheritance

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

Figure 33 shows class diagram AssetsOverview.



IdentifiedObject
AssetInfo
+AssetInfo
+AssetModel
0..1
0..1
IdentifiedObject
AssetModel
+AssetInfo
0..1
IdentifiedObject
AssetFunction
+ programID: String [0..1]
+ firmwareID: String [0..1]
+ hardwareID: String [0..1]
+ password: String [0..1]
+ configID: String [0..1]
ProductAssetModel
+ modelNumber: String [0..1]
+ modelVersion: String [0..1]
+ corporateStandardKind: CorporateStandardKind [0..1]
+ usageKind: AssetModelUsageKind [0..1]
+ weightTotal: Weight [0..1]
+ProductAssetModel
0..*
+Manufacturer
0..1
OrganisationRole
Manufacturer
+Assets
0..*
IdentifiedObject
Asset
+ type: String [0..1]
+ utcNumber: String [0..1]
+ serialNumber: String [0..1]
+ lotNumber: String [0..1]
+ purchasePrice: Money [0..1]
+ critical: Boolean [0..1]
+ electronicAddress: ElectronicAddress [0..1]
+ lifecycle: LifecycleDate [0..1]
+ acceptanceTest: AcceptanceTest [0..1]
+ initialCondition: String [0..1]
+ initialLossOfLife: PerCent [0..1]
+ status: Status [0..1]
+Assets
0..*
+OrganisationRoles
0..*
OrganisationRole
AssetOrganisationRole
AssetOwner
+Assets
0..*
ComMedia
0..1
+AssetContainer
AssetContainer
+AssetContainer
0..1
+Seals
0..*
IdentifiedObject
Seal
+ sealNumber: String [0..1]
+ kind: SealKind [0..1]
+ condition: SealConditionKind [0..1]
+ appliedDateTime: DateTime [0..1]
IEC 309/13

Figure 33 – Class diagram Assets::AssetsOverview

This diagram shows normative classes from this package.

6.4.2 AcceptanceTest compound

Acceptance test for assets.

Table 37 shows all attributes of AcceptanceTest.

Table 37 – Attributes of Assets::AcceptanceTest

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

6.4.3 LifecycleDate compound

Dates for lifecycle events of an asset.

Table 38 shows all attributes of LifecycleDate.

Table 38 – Attributes of Assets::LifecycleDate

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

6.4.4 AssetModelUsageKind enumeration

Usage for an asset model.

Table 39 shows all literals of AssetModelUsageKind.

Table 39 – Literals of Assets::AssetModelUsageKind

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

6.4.5 CorporateStandardKind enumeration

Kind of corporate standard.

Table 40 shows all literals of CorporateStandardKind.

Table 40 – Literals of Assets::CorporateStandardKind

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

6.4.6 SealConditionKind enumeration

Kind of seal condition.

Table 41 shows all literals of SealConditionKind.

Table 41 – Literals of Assets::SealConditionKind

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

6.4.7 SealKind enumeration

Kind of seal.

Table 42 shows all literals of SealKind.

Table 42 – Literals of Assets::SealKind

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

6.4.8 Asset

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

Table 43 shows all attributes of Asset.

Table 43 – Attributes of Assets::Asset

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

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

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

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

6.4.9 AssetContainer

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

Table 45 shows all attributes of AssetContainer.

Table 45 – Attributes of Assets::AssetContainer

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

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

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

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

6.4.10 AssetFunction

Function performed by an asset.

Table 47 shows all attributes of AssetFunction.

Table 47 – Attributes of Assets::AssetFunction

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

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

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

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

6.4.11 AssetInfo

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

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

Table 49 shows all attributes of AssetInfo.

Table 49 – Attributes of Assets::AssetInfo

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

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

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

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

6.4.12 AssetModel

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

Table 51 shows all attributes of AssetModel.

Table 51 – Attributes of Assets::AssetModel

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

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

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

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

6.4.13 AssetOrganisationRole

Role an organisation plays with respect to asset.

Table 53 shows all attributes of AssetOrganisationRole.

Table 53 – Attributes of Assets::AssetOrganisationRole

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

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

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

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

6.4.14 AssetOwner

Owner of the asset.

Table 55 shows all attributes of AssetOwner.

Table 55 – Attributes of Assets::AssetOwner

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

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

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

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

6.4.15 ComMedia

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

Table 57 shows all attributes of ComMedia.

Table 57 – Attributes of Assets::ComMedia

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

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

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

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

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

6.4.16 Manufacturer

Organisation that manufactures asset products.

Table 59 shows all attributes of Manufacturer.

Table 59 – Attributes of Assets::Manufacturer

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

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

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

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

6.4.17 ProductAssetModel

Asset model by a specific manufacturer.

Table 61 shows all attributes of ProductAssetModel.

Table 61 – Attributes of Assets::ProductAssetModel

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

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

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

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

6.4.18 Seal

Physically controls access to AssetContainers.

Table 63 shows all attributes of Seal.

Table 63 – Attributes of Assets::Seal

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

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

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

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

6.5 Package AssetInfo

6.5.1 General

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

Figure 34 shows class diagram AssetInfoInheritance.

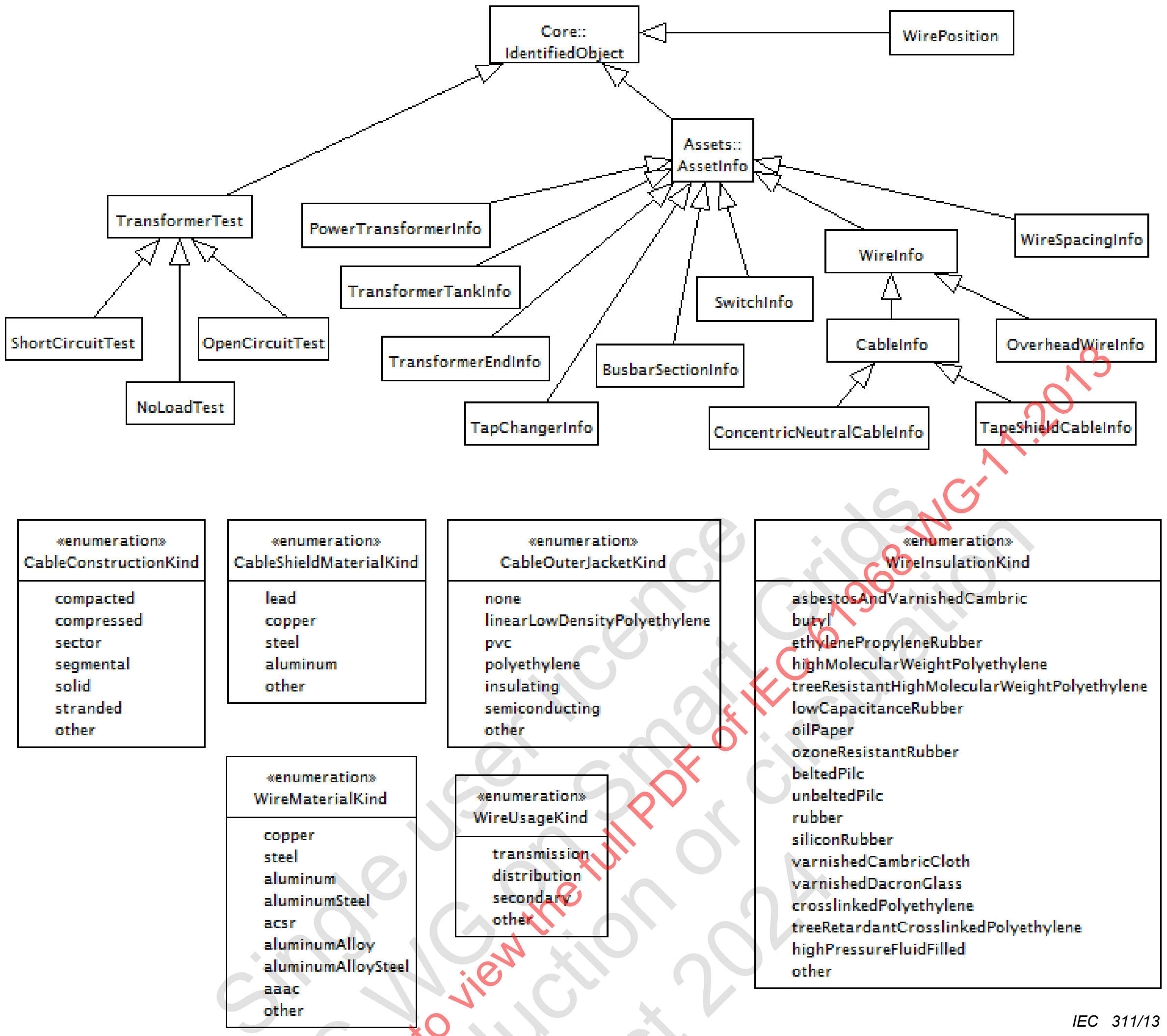


Figure 34 – Class diagram AssetInfo::AssetInfoInheritance

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

Figure 35 shows class diagram AssetInfoOverview.

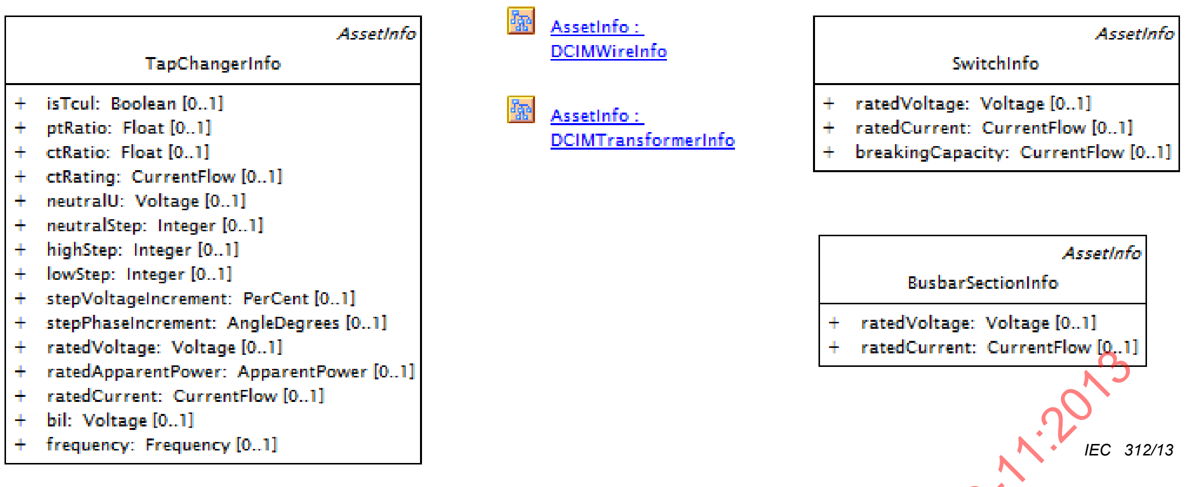
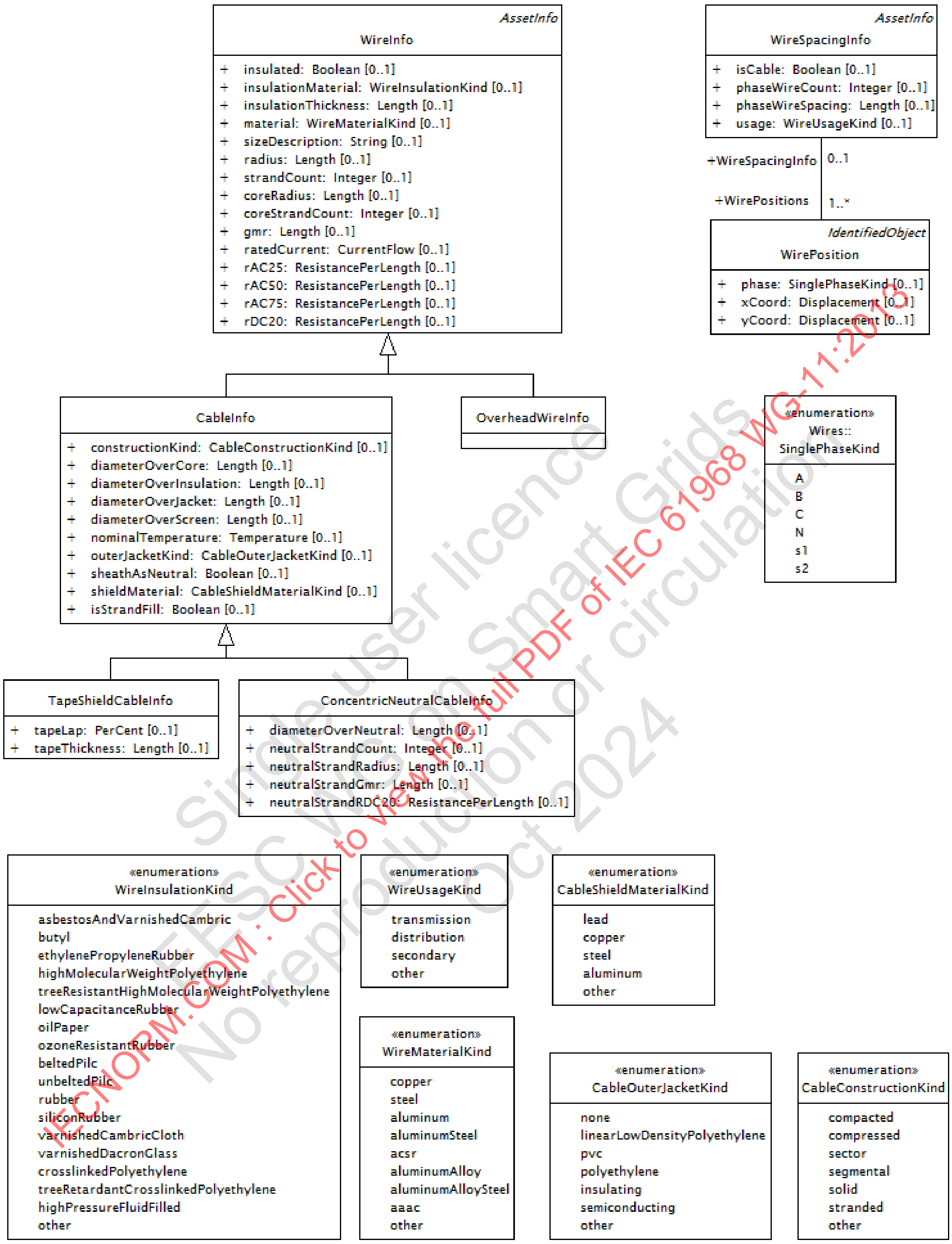

AssetInfo
TapChangerInfo
+ isTcul: Boolean [0..1]
+ ptRatio: Float [0..1]
+ ctRatio: Float [0..1]
+ ctRating: CurrentFlow [0..1]
+ neutralU: Voltage [0..1]
+ neutralStep: Integer [0..1]
+ highStep: Integer [0..1]
+ lowStep: Integer [0..1]
+ stepVoltageIncrement: PerCent [0..1]
+ stepPhaseIncrement: AngleDegrees [0..1]
+ ratedVoltage: Voltage [0..1]
+ ratedApparentPower: ApparentPower [0..1]
+ ratedCurrent: CurrentFlow [0..1]
+ bil: Voltage [0..1]
+ frequency: Frequency [0..1]
AssetInfo : DCIMWireInfo
AssetInfo : DCIMTransformerInfo
AssetInfo
SwitchInfo
+ ratedVoltage: Voltage [0..1]
+ ratedCurrent: CurrentFlow [0..1]
+ breakingCapacity: CurrentFlow [0..1]
AssetInfo
BusbarSectionInfo
+ ratedVoltage: Voltage [0..1]
+ ratedCurrent: CurrentFlow [0..1]
IEC 312/13

Figure 35 – Class diagram AssetInfo::AssetInfoOverview

This diagram shows normative classes from this package.

Figure 36 shows class diagram DCIMWireInfo.


AssetInfo
WireInfo
+ insulated: Boolean [0..1]
+ insulationMaterial: WireInsulationKind [0..1]
+ insulationThickness: Length [0..1]
+ material: WireMaterialKind [0..1]
+ sizeDescription: String [0..1]
+ radius: Length [0..1]
+ strandCount: Integer [0..1]
+ coreRadius: Length [0..1]
+ coreStrandCount: Integer [0..1]
+ gmr: Length [0..1]
+ ratedCurrent: CurrentFlow [0..1]
+ rAC25: ResistancePerLength [0..1]
+ rAC50: ResistancePerLength [0..1]
+ rAC75: ResistancePerLength [0..1]
+ rDC20: ResistancePerLength [0..1]
AssetInfo
WireSpacingInfo
+ isCable: Boolean [0..1]
+ phaseWireCount: Integer [0..1]
+ phaseWireSpacing: Length [0..1]
+ usage: WireUsageKind [0..1]
+WireSpacingInfo
0..1
+WirePositions
1..*
IdentifiedObject
WirePosition
+ phase: SinglePhaseKind [0..1]
+ xCoord: Displacement [0..1]
+ yCoord: Displacement [0..1]
CableInfo
+ constructionKind: CableConstructionKind [0..1]
+ diameterOverCore: Length [0..1]
+ diameterOverInsulation: Length [0..1]
+ diameterOverJacket: Length [0..1]
+ diameterOverScreen: Length [0..1]
+ nominalTemperature: Temperature [0..1]
+ outerJacketKind: CableOuterJacketKind [0..1]
+ sheathAsNeutral: Boolean [0..1]
+ shieldMaterial: CableShieldMaterialKind [0..1]
+ isStrandFill: Boolean [0..1]
OverheadWireInfo
«enumeration»
Wires::
SinglePhaseKind
A
B
C
N
s1
s2
TapeShieldCableInfo
+ tapeLap: PerCent [0..1]
+ tapeThickness: Length [0..1]
ConcentricNeutralCableInfo
+ diameterOverNeutral: Length [0..1]
+ neutralStrandCount: Integer [0..1]
+ neutralStrandRadius: Length [0..1]
+ neutralStrandGmr: Length [0..1]
+ neutralStrandRDC20: ResistancePerLength [0..1]
«enumeration»
WireInsulationKind
asbestosAndVarnishedCambric
butyl
ethylenePropyleneRubber
highMolecularWeightPolyethylene
treeResistantHighMolecularWeightPolyethylene
lowCapacitanceRubber
oilPaper
ozoneResistantRubber
beltedPilc
unbeltedPilc
rubber
siliconRubber
varnishedCambricCloth
varnishedDacronGlass
crosslinkedPolyethylene
treeRetardantCrosslinkedPolyethylene
highPressureFluidFilled
other
«enumeration»
WireUsageKind
transmission
distribution
secondary
other
«enumeration»
CableShieldMaterialKind
lead
copper
steel
aluminum
other
«enumeration»
WireMaterialKind
copper
steel
aluminum
aluminumSteel
acsr
aluminumAlloy
aluminumAlloySteel
aaac
other
«enumeration»
CableOuterJacketKind
none
linearLowDensityPolyethylene
pvc
polyethylene
insulating
semiconducting
other
«enumeration»
CableConstructionKind
compacted
compressed
sector
segmental
solid
stranded
other

IEC 313/13

Figure 36 – Class diagram AssetInfo::DCIMWireInfo

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

Figure 37 shows class diagram DCIMTransformerInfo.

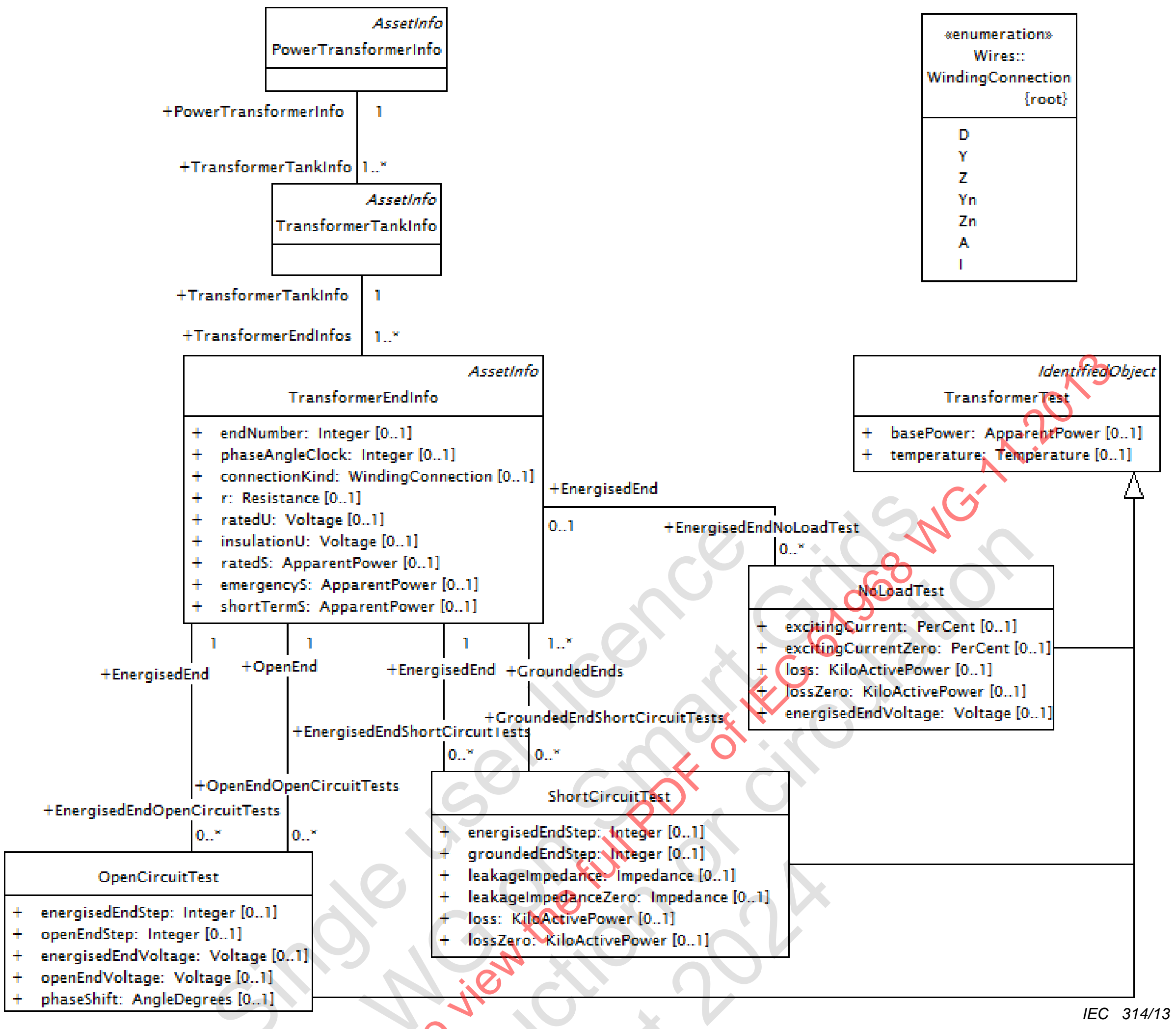

AssetInfo
PowerTransformerInfo
+PowerTransformerInfo 1
+TransformerTankInfo 1..*
AssetInfo
TransformerTankInfo
+TransformerTankInfo 1
+TransformerEndInfos 1..*
AssetInfo
TransformerEndInfo
+ endNumber: Integer [0..1]
+ phaseAngleClock: Integer [0..1]
+ connectionKind: WindingConnection [0..1]
+ r: Resistance [0..1]
+ ratedU: Voltage [0..1]
+ insulationU: Voltage [0..1]
+ ratedS: ApparentPower [0..1]
+ emergencyS: ApparentPower [0..1]
+ shortTermS: ApparentPower [0..1]
«enumeration»
Wires::
WindingConnection
{root}
D
Y
Z
Yn
Zn
A
I
IdentifiedObject
TransformerTest
+ basePower: ApparentPower [0..1]
+ temperature: Temperature [0..1]
+EnergisedEnd 0..1
+EnergisedEndNoLoadTest 0..*
NoLoadTest
+ excitingCurrent: PerCent [0..1]
+ excitingCurrentZero: PerCent [0..1]
+ loss: KiloActivePower [0..1]
+ lossZero: KiloActivePower [0..1]
+ energisedEndVoltage: Voltage [0..1]
+EnergisedEnd 1
+OpenEnd 1
+EnergisedEnd 1
+GroundedEnds 1..*
+GroundedEndShortCircuitTests 0..*
+EnergisedEndShortCircuitTests 0..*
+OpenEndOpenCircuitTests 0..*
+EnergisedEndOpenCircuitTests 0..*
ShortCircuitTest
+ energisedEndStep: Integer [0..1]
+ groundedEndStep: Integer [0..1]
+ leakageImpedance: Impedance [0..1]
+ leakageImpedanceZero: Impedance [0..1]
+ loss: KiloActivePower [0..1]
+ lossZero: KiloActivePower [0..1]
OpenCircuitTest
+ energisedEndStep: Integer [0..1]
+ openEndStep: Integer [0..1]
+ energisedEndVoltage: Voltage [0..1]
+ openEndVoltage: Voltage [0..1]
+ phaseShift: AngleDegrees [0..1]
IEC 314/13

Figure 37 – Class diagram AssetInfo::DCIMTransformerInfo

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

6.5.2 BusbarSectionInfo

Busbar section data.

Table 65 shows all attributes of BusbarSectionInfo.

Table 65 – Attributes of AssetInfo::BusbarSectionInfo

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

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

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

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

6.5.3 CableConstructionKind enumeration

Kind of cable construction.

Table 67 shows all literals of CableConstructionKind.

Table 67 – Literals of AssetInfo::CableConstructionKind

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

6.5.4 CableInfo

Cable data.

Table 68 shows all attributes of CableInfo.

Table 68 – Attributes of AssetInfo::CableInfo

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

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

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

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

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

6.5.5 CableOuterJacketKind enumeration

Kind of cable outer jacket.

Table 70 shows all literals of CableOuterJacketKind.

Table 70 – Literals of AssetInfo::CableOuterJacketKind

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

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

6.5.6 CableShieldMaterialKind enumeration

Kind of cable shield material.

Table 71 shows all literals of CableShieldMaterialKind.

Table 71 – Literals of AssetInfo::CableShieldMaterialKind

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

6.5.7 ConcentricNeutralCableInfo

Concentric neutral cable data.

Table 72 shows all attributes of ConcentricNeutralCableInfo.

Table 72 – Attributes of AssetInfo::ConcentricNeutralCableInfo

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

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

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

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

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

6.5.8 NoLoadTest

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

Table 74 shows all attributes of NoLoadTest.

Table 74 – Attributes of AssetInfo::NoLoadTest

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

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

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

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

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

6.5.9 OpenCircuitTest

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

Table 76 shows all attributes of OpenCircuitTest.

Table 76 – Attributes of AssetInfo::OpenCircuitTest

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

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

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

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

6.5.10 OverheadWireInfo

Overhead wire data.

Table 78 shows all attributes of OverheadWireInfo.

Table 78 – Attributes of AssetInfo::OverheadWireInfo

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

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

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

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

6.5.11 PowerTransformerInfo

Set of power transformer data, from an equipment library.

Table 80 shows all attributes of PowerTransformerInfo.

Table 80 – Attributes of AssetInfo::PowerTransformerInfo

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

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

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

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

6.5.12 ShortCircuitTest

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

Table 82 shows all attributes of ShortCircuitTest.

Table 82 – Attributes of AssetInfo::ShortCircuitTest

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

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

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

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

6.5.13 SwitchInfo

Switch data.

Table 84 shows all attributes of SwitchInfo.

Table 84 – Attributes of AssetInfo::SwitchInfo

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

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

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

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

6.5.14 TapChangerInfo

Tap changer data.

Table 86 shows all attributes of TapChangerInfo.

Table 86 – Attributes of AssetInfo::TapChangerInfo

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

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

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

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

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

6.5.15 TapeShieldCableInfo

Tape shield cable data.

Table 88 shows all attributes of TapeShieldCableInfo.

Table 88 – Attributes of AssetInfo::TapeShieldCableInfo

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

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

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

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

6.5.16 TransformerEndInfo

Transformer end data.

Table 90 shows all attributes of TransformerEndInfo.

Table 90 – Attributes of AssetInfo::TransformerEndInfo

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

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

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

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

6.5.17 TransformerTankInfo

Set of transformer tank data, from an equipment library.

Table 92 shows all attributes of TransformerTankInfo.

Table 92 – Attributes of AssetInfo::TransformerTankInfo

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

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

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

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

6.5.18 TransformerTest

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

Table 94 shows all attributes of TransformerTest.

Table 94 – Attributes of AssetInfo::TransformerTest

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

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

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

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

6.5.19 WireInfo

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

Table 96 shows all attributes of WireInfo.

Table 96 – Attributes of AssetInfo::WireInfo

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

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

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

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

6.5.20 WireInsulationKind enumeration

Kind of wire insulation.

Table 98 shows all literals of WireInsulationKind.

Table 98 – Literals of AssetInfo::WireInsulationKind

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

6.5.21 WireMaterialKind enumeration

Kind of wire material.

Table 99 shows all literals of WireMaterialKind.

Table 99 – Literals of AssetInfo::WireMaterialKind

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

6.5.22 WirePosition

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

Table 100 shows all attributes of WirePosition.

Table 100 – Attributes of AssetInfo::WirePosition

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

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

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

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

6.5.23 WireSpacingInfo

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

Table 102 shows all attributes of WireSpacingInfo.

Table 102 – Attributes of AssetInfo::WireSpacingInfo

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

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

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

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

6.5.24 WireUsageKind enumeration

Kind of wire usage.

Table 104 shows all literals of WireUsageKind.

Table 104 – Literals of AssetInfo::WireUsageKind

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

6.6 Package Work

6.6.1 General

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

Figure 38 shows class diagram WorkInheritance.

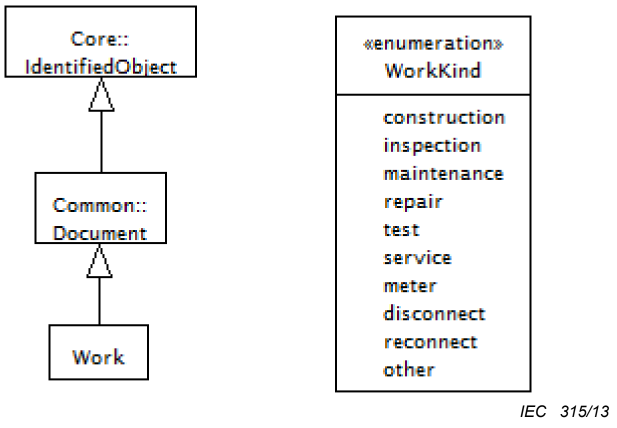


Figure 38 – Class diagram Work::WorkInheritance

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

Figure 39 shows class diagram WorkOverview.

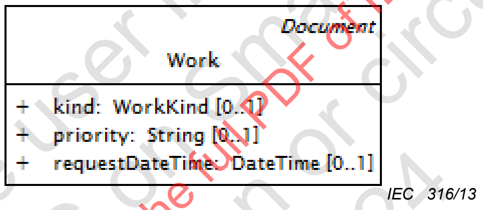


Figure 39 – Class diagram Work::WorkOverview

This diagram shows normative classes from this package.

6.6.2 WorkKind enumeration

Kind of work.

Table 105 shows all literals of WorkKind.

Table 105 – Literals of Work::WorkKind

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

6.6.3 Work

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

Table 106 shows all attributes of Work.

Table 106 – Attributes of Work::Work

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

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

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

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

6.7 Package Customers

6.7.1 General

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

Figure 40 shows class diagram CustomersInheritance.

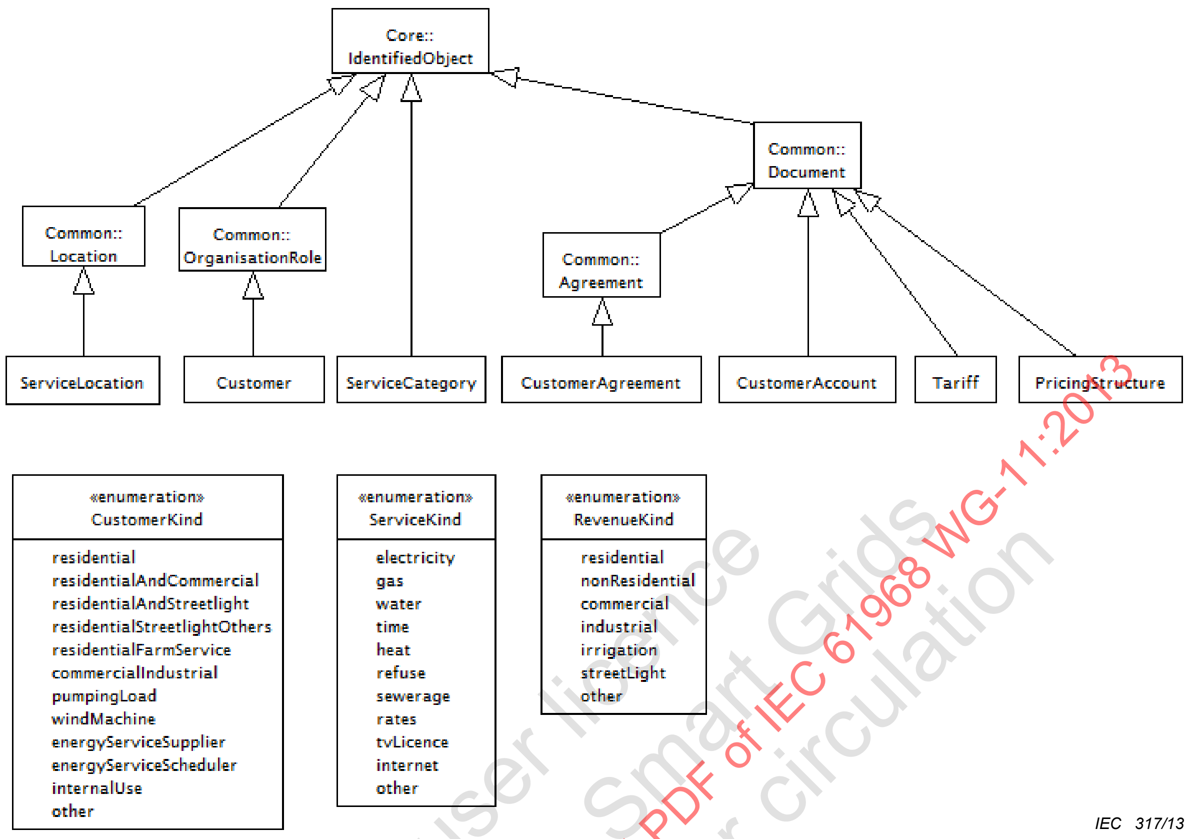


Figure 40 – Class diagram Customers::CustomersInheritance

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

Figure 41 shows class diagram CustomersOverview.

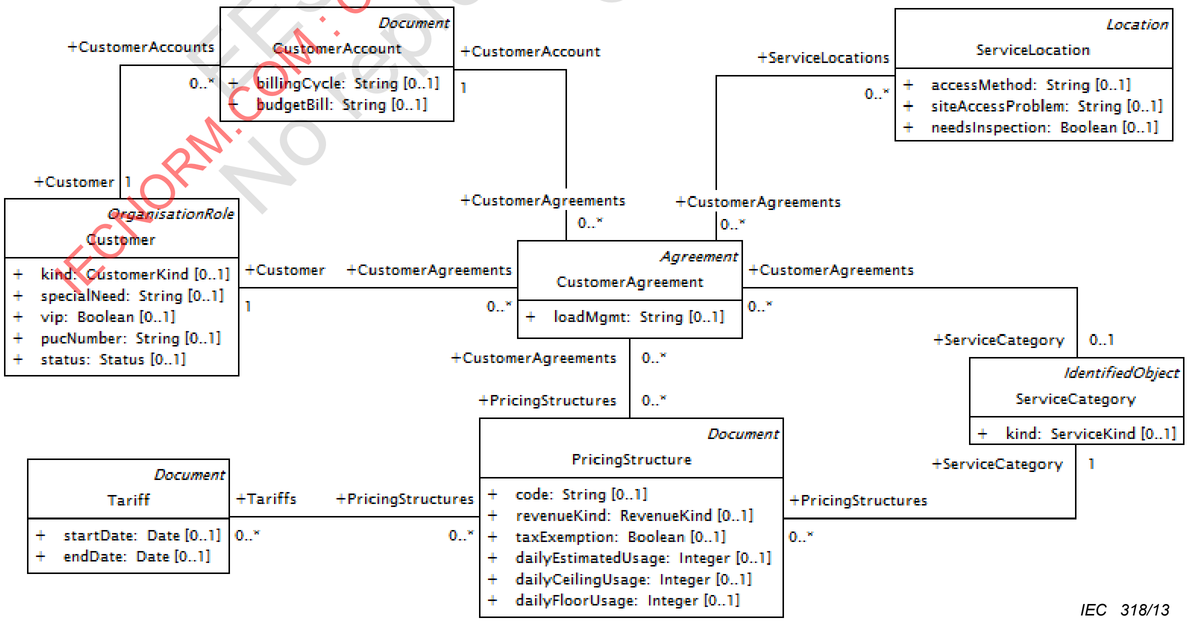


Figure 41 – Class diagram Customers::CustomersOverview

This diagram shows normative classes from this package.

6.7.2 CustomerKind enumeration

Kind of customer.

Table 108 shows all literals of CustomerKind.

Table 108 – Literals of Customers::CustomerKind

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

6.7.3 RevenueKind enumeration

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

Table 109 shows all literals of RevenueKind.

Table 109 – Literals of Customers::RevenueKind

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

6.7.4 ServiceKind enumeration

Kind of service.

Table 110 shows all literals of ServiceKind.

Table 110 – Literals of Customers::ServiceKind

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

6.7.5 Customer

Organisation receiving services from service supplier.

Table 111 shows all attributes of Customer.

Table 111 – Attributes of Customers::Customer

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

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

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

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

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

6.7.6 CustomerAccount

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

Table 113 shows all attributes of CustomerAccount.

Table 113 – Attributes of Customers::CustomerAccount

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

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

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

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

6.7.7 CustomerAgreement

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

Table 115 shows all attributes of CustomerAgreement.

Table 115 – Attributes of Customers::CustomerAgreement

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

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

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

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

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

6.7.8 PricingStructure

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

Table 117 shows all attributes of PricingStructure.

Table 117 – Attributes of Customers::PricingStructure

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

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

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

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

6.7.9 ServiceCategory

Category of service provided to the customer.

Table 119 shows all attributes of ServiceCategory.

Table 119 – Attributes of Customers::ServiceCategory

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

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

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

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

6.7.10 ServiceLocation

A real estate location, commonly referred to as premises.

Table 121 shows all attributes of ServiceLocation.

Table 121 – Attributes of Customers::ServiceLocation

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

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

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

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

6.7.11 Tariff

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

Table 123 shows all attributes of Tariff.

Table 123 – Attributes of Customers::Tariff

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

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

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

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

6.8 Package Metering

6.8.1 General

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

Figure 42 shows class diagram MeteringInheritance.

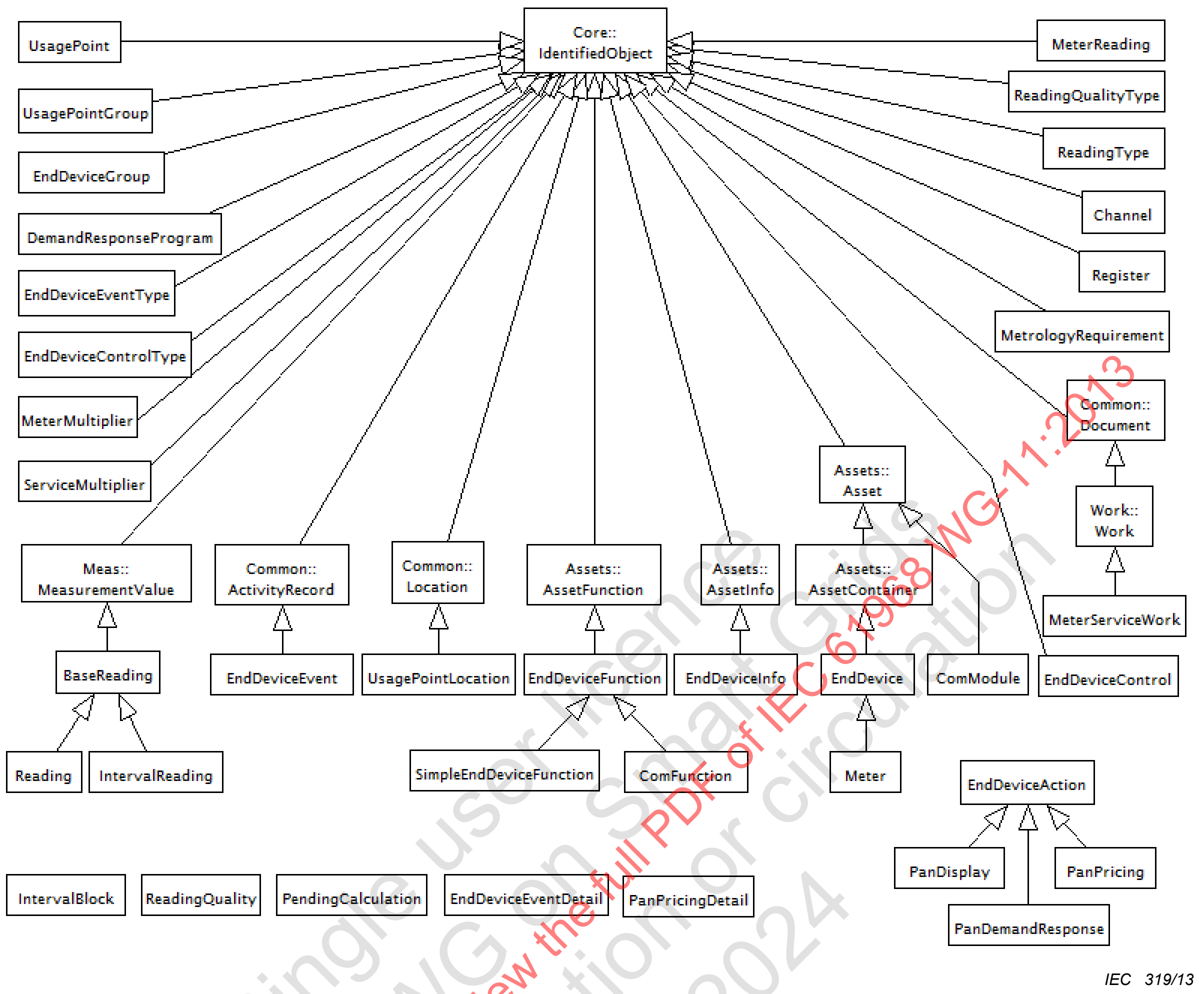
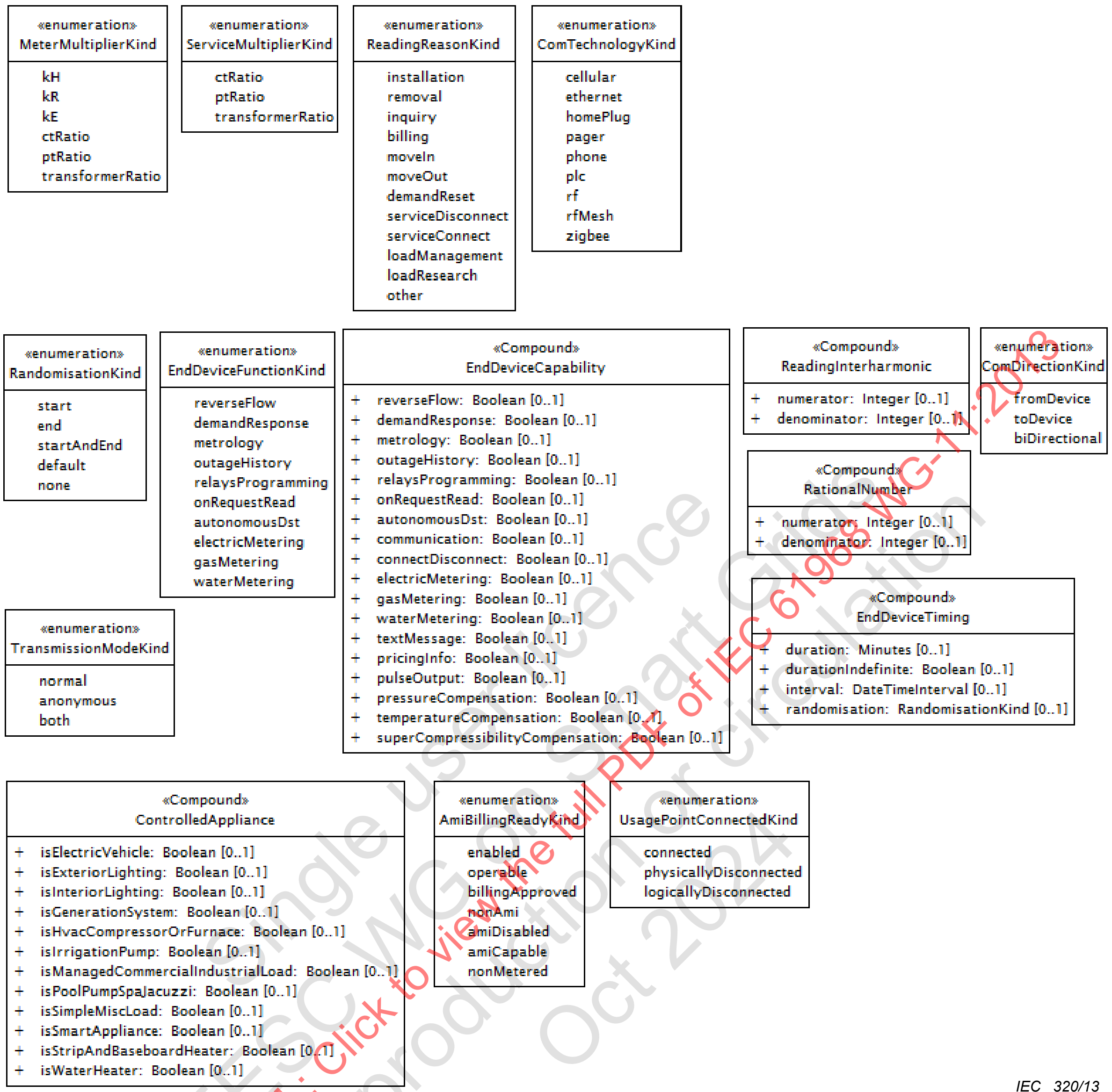


Figure 42 – Class diagram Metering::MeteringInheritance

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

Figure 43 shows class diagram MeteringDatatypes.


«enumeration»
MeterMultiplierKind
kH
kR
kE
ctRatio
ptRatio
transformerRatio
«enumeration»
ServiceMultiplierKind
ctRatio
ptRatio
transformerRatio
«enumeration»
ReadingReasonKind
installation
removal
inquiry
billing
moveIn
moveOut
demandReset
serviceDisconnect
serviceConnect
loadManagement
loadResearch
other
«enumeration»
ComTechnologyKind
cellular
ethernet
homePlug
pager
phone
plc
rf
rfMesh
zigbee
«enumeration»
RandomisationKind
start
end
startAndEnd
default
none
«enumeration»
EndDeviceFunctionKind
reverseFlow
demandResponse
metrology
outageHistory
relaysProgramming
onRequestRead
autonomousDst
electricMetering
gasMetering
waterMetering
«Compound»
EndDeviceCapability
+ reverseFlow: Boolean [0..1]
+ demandResponse: Boolean [0..1]
+ metrology: Boolean [0..1]
+ outageHistory: Boolean [0..1]
+ relaysProgramming: Boolean [0..1]
+ onRequestRead: Boolean [0..1]
+ autonomousDst: Boolean [0..1]
+ communication: Boolean [0..1]
+ connectDisconnect: Boolean [0..1]
+ electricMetering: Boolean [0..1]
+ gasMetering: Boolean [0..1]
+ waterMetering: Boolean [0..1]
+ textMessage: Boolean [0..1]
+ pricingInfo: Boolean [0..1]
+ pulseOutput: Boolean [0..1]
+ pressureCompensation: Boolean [0..1]
+ temperatureCompensation: Boolean [0..1]
+ superCompressibilityCompensation: Boolean [0..1]
«Compound»
ReadingInterharmonic
+ numerator: Integer [0..1]
+ denominator: Integer [0..1]
«enumeration»
ComDirectionKind
fromDevice
toDevice
biDirectional
«Compound»
RationalNumber
+ numerator: Integer [0..1]
+ denominator: Integer [0..1]
«enumeration»
TransmissionModeKind
normal
anonymous
both
«Compound»
EndDeviceTiming
+ duration: Minutes [0..1]
+ durationIndefinite: Boolean [0..1]
+ interval: DateTimeInterval [0..1]
+ randomisation: RandomisationKind [0..1]
«Compound»
ControlledAppliance
+ isElectricVehicle: Boolean [0..1]
+ isExteriorLighting: Boolean [0..1]
+ isInteriorLighting: Boolean [0..1]
+ isGenerationSystem: Boolean [0..1]
+ isHvacCompressorOrFurnace: Boolean [0..1]
+ isIrrigationPump: Boolean [0..1]
+ isManagedCommercialIndustrialLoad: Boolean [0..1]
+ isPoolPumpSpaJacuzzi: Boolean [0..1]
+ isSimpleMiscLoad: Boolean [0..1]
+ isSmartAppliance: Boolean [0..1]
+ isStripAndBaseboardHeater: Boolean [0..1]
+ isWaterHeater: Boolean [0..1]
«enumeration»
AmiBillingReadyKind
enabled
operable
billingApproved
nonAmi
amiDisabled
amiCapable
nonMetered
«enumeration»
UsagePointConnectedKind
connected
physicallyDisconnected
logicallyDisconnected

IEC 320/13

Figure 43 – Class diagram Metering::MeteringDatatypes

This diagram shows enumerations and compound types from this package.

Figure 44 shows class diagram MeteringOverviewShort.

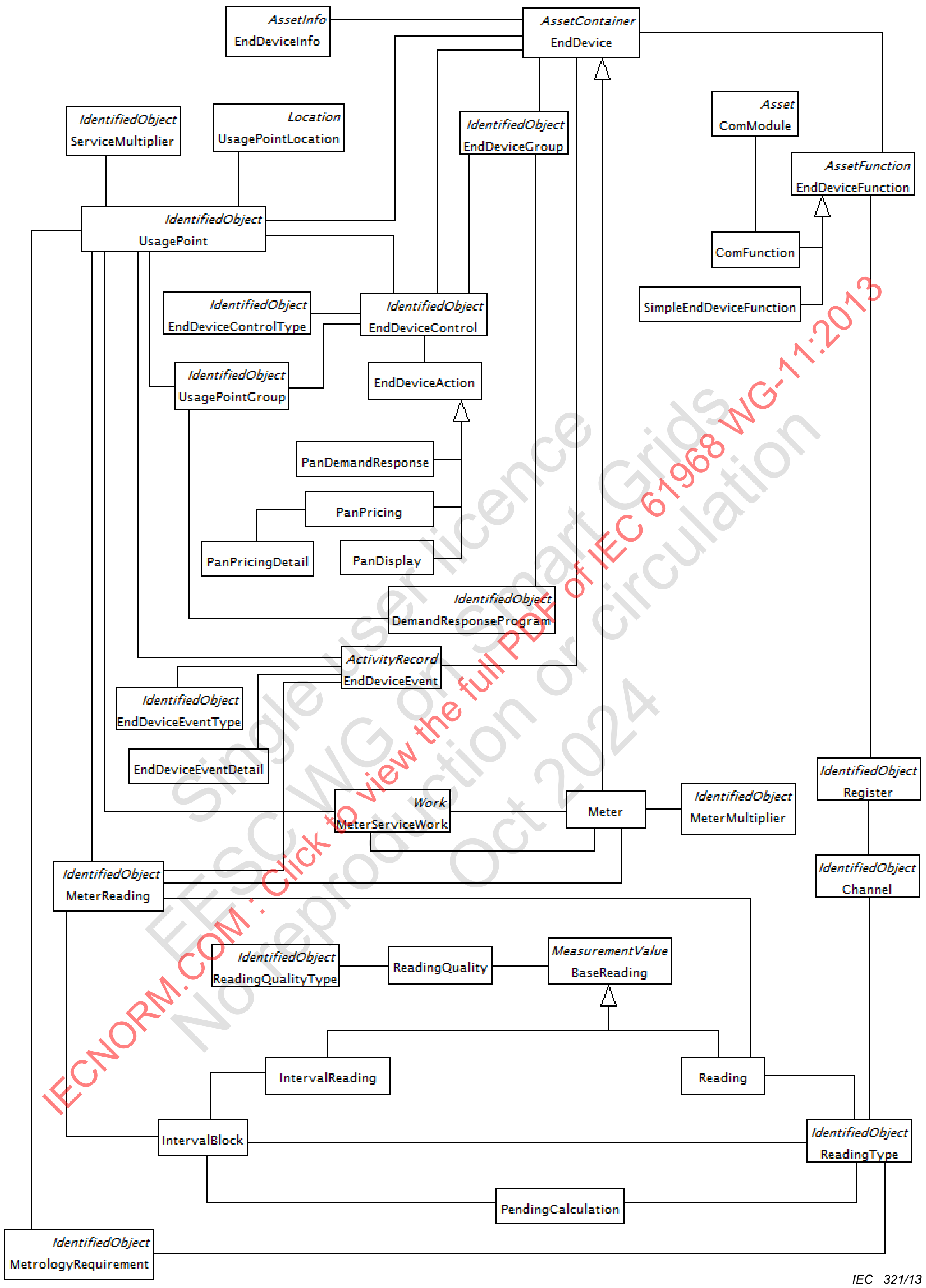


Figure 44 – Class diagram Metering::MeteringOverviewShort

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

Figure 45 shows class diagram MeteringUsagePoints.

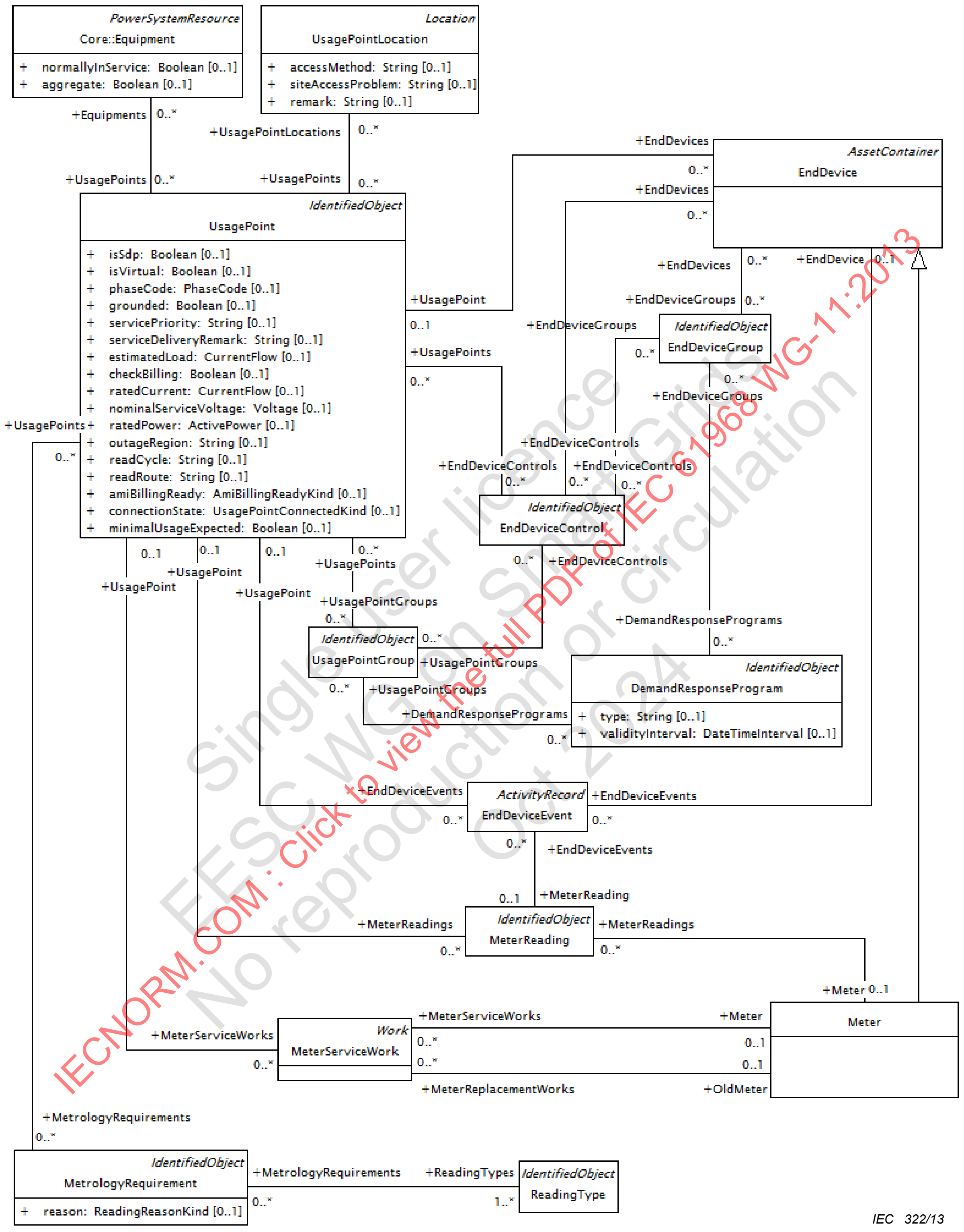

PowerSystemResource
Core::Equipment
+ normallyInService: Boolean [0..1]
+ aggregate: Boolean [0..1]
Location
UsagePointLocation
+ accessMethod: String [0..1]
+ siteAccessProblem: String [0..1]
+ remark: String [0..1]
AssetContainer
EndDevice
IdentifiedObject
UsagePoint
+ isSdp: Boolean [0..1]
+ isVirtual: Boolean [0..1]
+ phaseCode: PhaseCode [0..1]
+ grounded: Boolean [0..1]
+ servicePriority: String [0..1]
+ serviceDeliveryRemark: String [0..1]
+ estimatedLoad: CurrentFlow [0..1]
+ checkBilling: Boolean [0..1]
+ ratedCurrent: CurrentFlow [0..1]
+ nominalServiceVoltage: Voltage [0..1]
+ ratedPower: ActivePower [0..1]
+ outageRegion: String [0..1]
+ readCycle: String [0..1]
+ readRoute: String [0..1]
+ amiBillingReady: AmiBillingReadyKind [0..1]
+ connectionState: UsagePointConnectedKind [0..1]
+ minimalUsageExpected: Boolean [0..1]
IdentifiedObject
EndDeviceGroup
IdentifiedObject
EndDeviceControl
IdentifiedObject
UsagePointGroup
IdentifiedObject
DemandResponseProgram
+ type: String [0..1]
+ validityInterval: DateTimeInterval [0..1]
ActivityRecord
EndDeviceEvent
IdentifiedObject
MeterReading
Work
MeterServiceWork
Meter
IdentifiedObject
MetrologyRequirement
+ reason: ReadingReasonKind [0..1]
IdentifiedObject
ReadingType
IEC 322/13

Figure 45 – Class diagram Metering::MeteringUsagePoints

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

Figure 46 shows class diagram MeteringEndDevices.

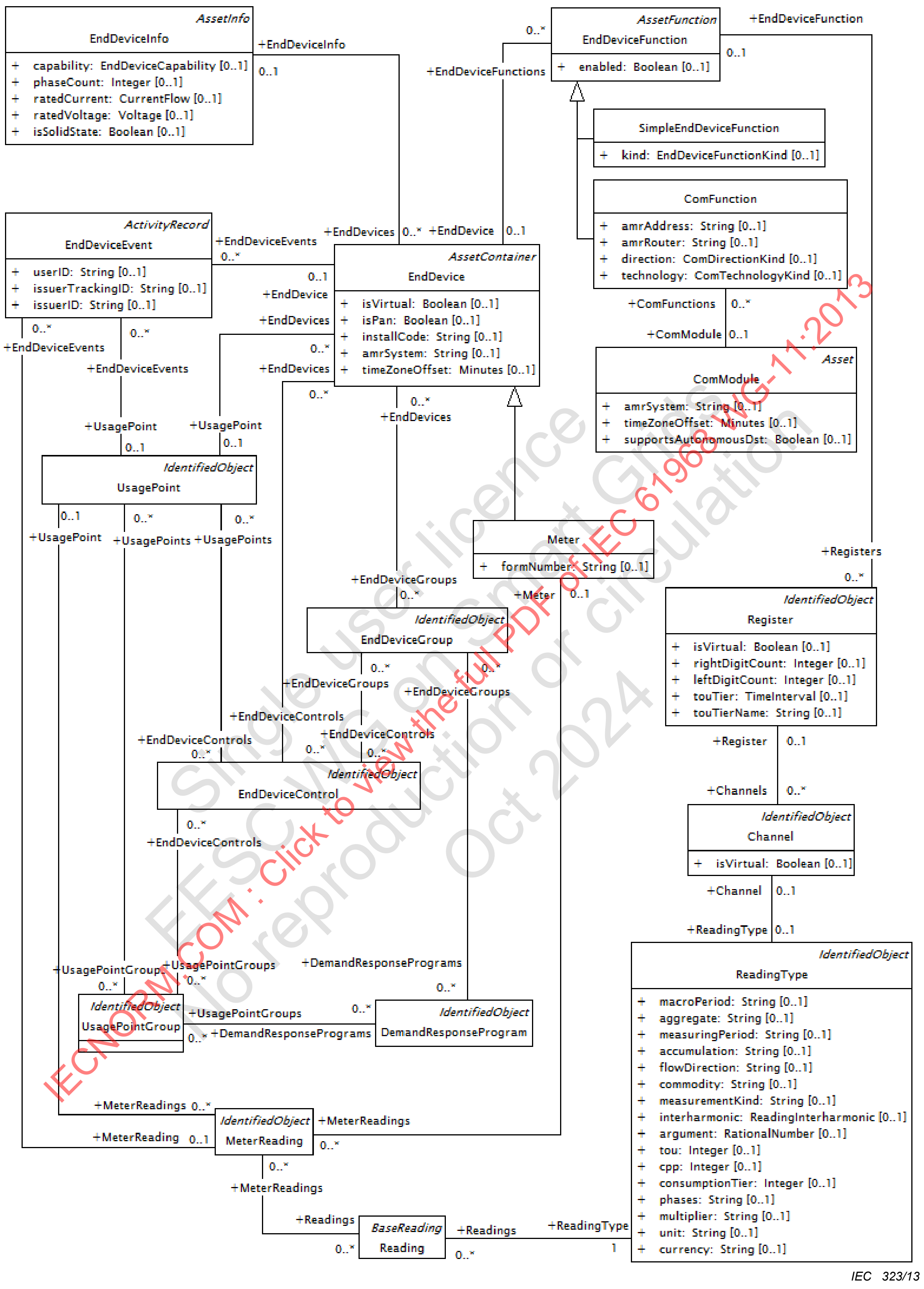

AssetInfo
EndDeviceInfo
+ capability: EndDeviceCapability [0..1]
+ phaseCount: Integer [0..1]
+ ratedCurrent: CurrentFlow [0..1]
+ ratedVoltage: Voltage [0..1]
+ isSolidState: Boolean [0..1]
AssetFunction
EndDeviceFunction
+ enabled: Boolean [0..1]
SimpleEndDeviceFunction
+ kind: EndDeviceFunctionKind [0..1]
ComFunction
+ amrAddress: String [0..1]
+ amrRouter: String [0..1]
+ direction: ComDirectionKind [0..1]
+ technology: ComTechnologyKind [0..1]
ActivityRecord
EndDeviceEvent
+ userID: String [0..1]
+ issuerTrackingID: String [0..1]
+ issuerID: String [0..1]
AssetContainer
EndDevice
+ isVirtual: Boolean [0..1]
+ isPan: Boolean [0..1]
+ installCode: String [0..1]
+ amrSystem: String [0..1]
+ timeZoneOffset: Minutes [0..1]
Asset
ComModule
+ amrSystem: String [0..1]
+ timeZoneOffset: Minutes [0..1]
+ supportsAutonomousDst: Boolean [0..1]
IdentifiedObject
UsagePoint
Meter
+ formNumber: String [0..1]
IdentifiedObject
Register
+ isVirtual: Boolean [0..1]
+ rightDigitCount: Integer [0..1]
+ leftDigitCount: Integer [0..1]
+ touTier: TimeInterval [0..1]
+ touTierName: String [0..1]
IdentifiedObject
EndDeviceGroup
IdentifiedObject
EndDeviceControl
IdentifiedObject
Channel
+ isVirtual: Boolean [0..1]
IdentifiedObject
ReadingType
+ macroPeriod: String [0..1]
+ aggregate: String [0..1]
+ measuringPeriod: String [0..1]
+ accumulation: String [0..1]
+ flowDirection: String [0..1]
+ commodity: String [0..1]
+ measurementKind: String [0..1]
+ interharmonic: ReadingInterharmonic [0..1]
+ argument: RationalNumber [0..1]
+ tou: Integer [0..1]
+ cpp: Integer [0..1]
+ consumptionTier: Integer [0..1]
+ phases: String [0..1]
+ multiplier: String [0..1]
+ unit: String [0..1]
+ currency: String [0..1]
IdentifiedObject
UsagePointGroup
IdentifiedObject
DemandResponseProgram
IdentifiedObject
MeterReading
BaseReading
Reading
+EndDeviceInfo 0..1
+EndDeviceFunctions 0..*
+EndDeviceFunction 0..1
+EndDevices 0..*
+EndDevice 0..1
+EndDeviceEvents 0..*
+EndDevice 0..1
+EndDevices 0..*
+EndDeviceEvents 0..*
+ComFunctions 0..*
+ComModule 0..1
+UsagePoint 0..1
+UsagePoints 0..*
+EndDeviceGroups 0..*
+Meter 0..1
+Registers 0..*
+Register 0..1
+Channels 0..*
+Channel 0..1
+ReadingType 0..1
+EndDeviceControls 0..*
+UsagePointGroups 0..*
+DemandResponsePrograms 0..*
+MeterReadings 0..*
+MeterReading 0..1
+Readings 0..*
+ReadingType 1
IEC 323/13

Figure 46 – Class diagram Metering::MeteringEndDevices

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

Figure 47 shows class diagram MeteringConfigurationEvents.

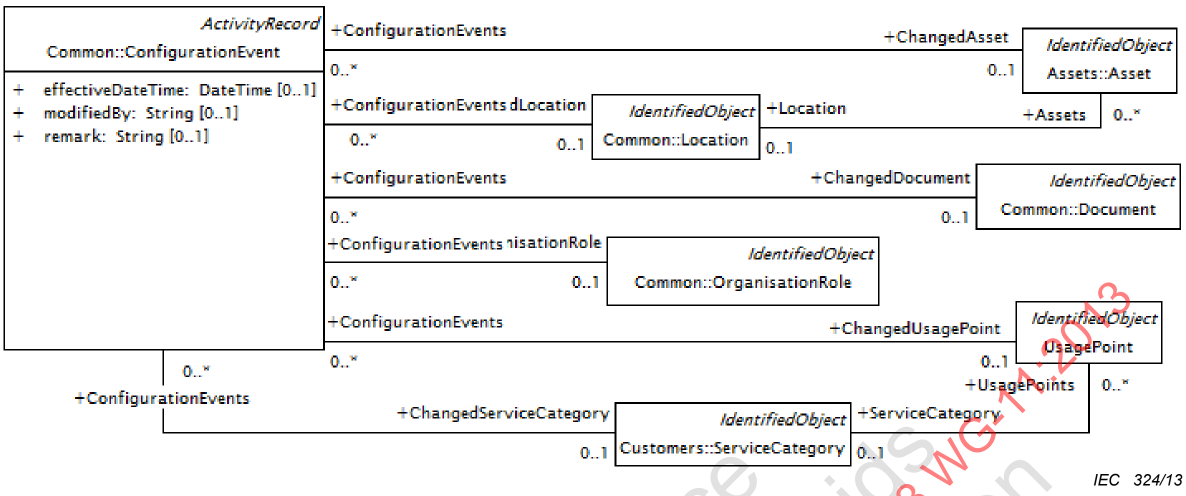


Figure 47 – Class diagram Metering::MeteringConfigurationEvents

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

Figure 48 shows class diagram MeteringMeterReadings.

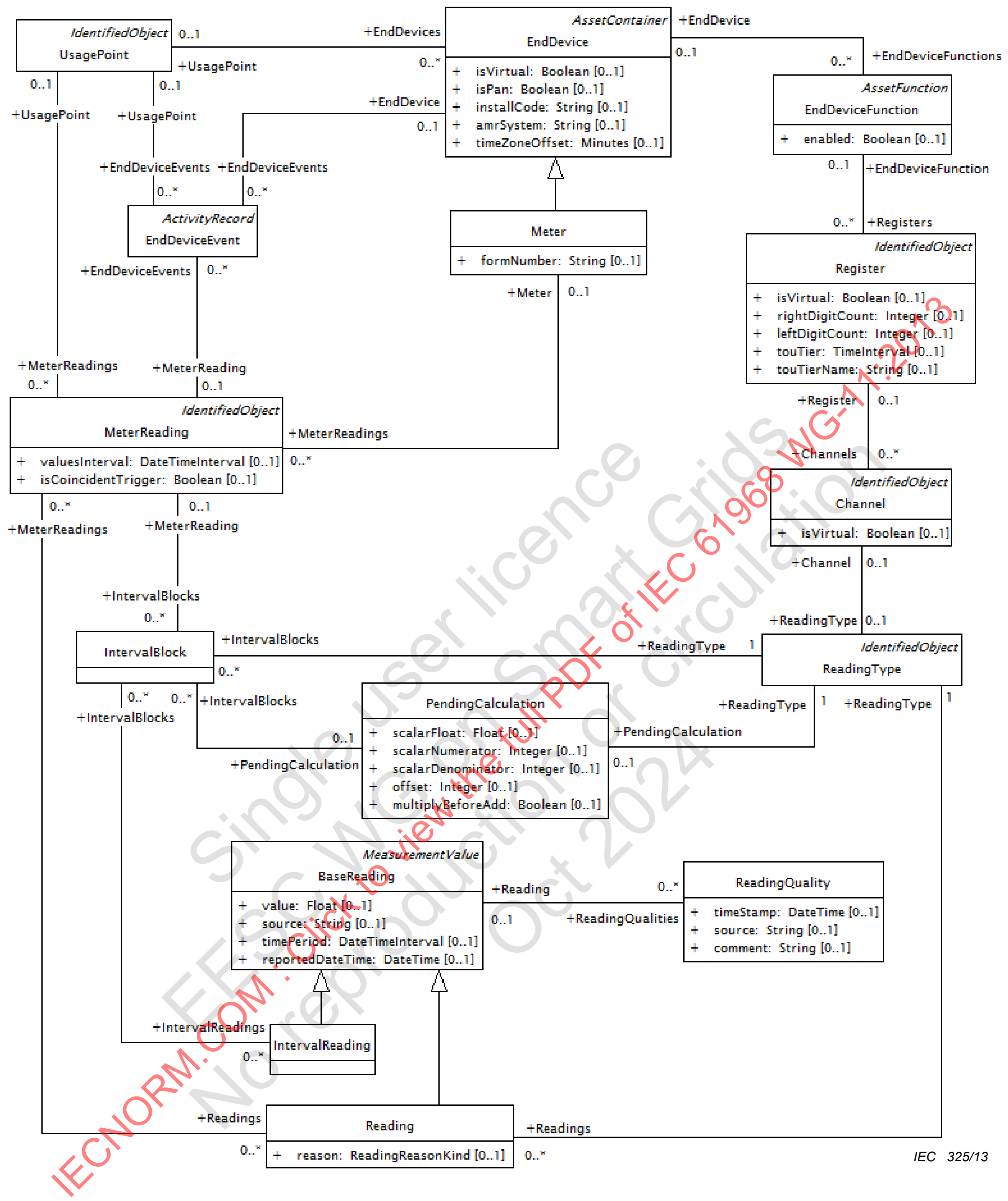


Figure 48 – Class diagram Metering::MeteringMeterReadings

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

Figure 49 shows class diagram MeteringEventsAndControls.

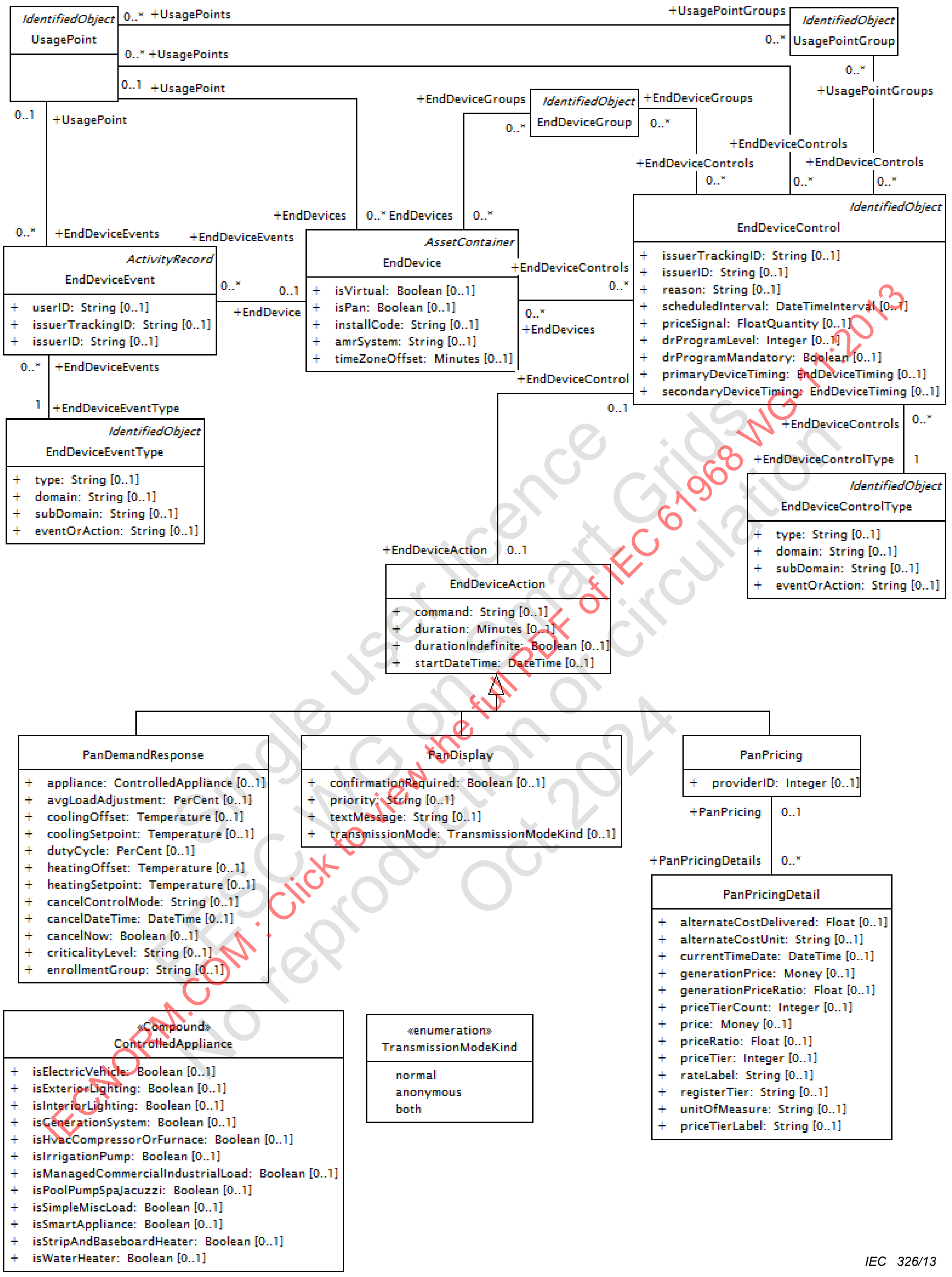

IdentifiedObject
UsagePoint
0..* +UsagePoints
+UsagePointGroups
IdentifiedObject
UsagePointGroup
0..* +UsagePoints
0..*
0..1 +UsagePoint
0..*
+UsagePointGroups
+EndDeviceGroups
IdentifiedObject
EndDeviceGroup
+EndDeviceGroups
0..1
+UsagePoint
0..*
0..*
+EndDeviceControls
+EndDeviceControls
+EndDeviceControls
0..*
0..*
0..*
IdentifiedObject
EndDeviceControl
+ issuerTrackingID: String [0..1]
+ issuerID: String [0..1]
+ reason: String [0..1]
+ scheduledInterval: DateTimeInterval [0..1]
+ priceSignal: FloatQuantity [0..1]
+ drProgramLevel: Integer [0..1]
+ drProgramMandatory: Boolean [0..1]
+ primaryDeviceTiming: EndDeviceTiming [0..1]
+ secondaryDeviceTiming: EndDeviceTiming [0..1]
+EndDevices
0..* EndDevices
0..*
0..*
+EndDeviceEvents
+EndDeviceEvents
AssetContainer
EndDevice
+ isVirtual: Boolean [0..1]
+ isPan: Boolean [0..1]
+ installCode: String [0..1]
+ amrSystem: String [0..1]
+ timeZoneOffset: Minutes [0..1]
+EndDeviceControls
0..*
0..*
+EndDevices
+EndDeviceControl
0..1
ActivityRecord
EndDeviceEvent
+ userID: String [0..1]
+ issuerTrackingID: String [0..1]
+ issuerID: String [0..1]
0..*
0..1
+EndDevice
0..*
+EndDeviceEvents
1
+EndDeviceEventType
IdentifiedObject
EndDeviceEventType
+ type: String [0..1]
+ domain: String [0..1]
+ subDomain: String [0..1]
+ eventOrAction: String [0..1]
+EndDeviceControls
0..*
+EndDeviceControlType
1
IdentifiedObject
EndDeviceControlType
+ type: String [0..1]
+ domain: String [0..1]
+ subDomain: String [0..1]
+ eventOrAction: String [0..1]
+EndDeviceAction
0..1
EndDeviceAction
+ command: String [0..1]
+ duration: Minutes [0..1]
+ durationIndefinite: Boolean [0..1]
+ startDateTime: DateTime [0..1]
PanDemandResponse
+ appliance: ControlledAppliance [0..1]
+ avgLoadAdjustment: PerCent [0..1]
+ coolingOffset: Temperature [0..1]
+ coolingSetpoint: Temperature [0..1]
+ dutyCycle: PerCent [0..1]
+ heatingOffset: Temperature [0..1]
+ heatingSetpoint: Temperature [0..1]
+ cancelControlMode: String [0..1]
+ cancelDateTime: DateTime [0..1]
+ cancelNow: Boolean [0..1]
+ criticalityLevel: String [0..1]
+ enrollmentGroup: String [0..1]
PanDisplay
+ confirmationRequired: Boolean [0..1]
+ priority: String [0..1]
+ textMessage: String [0..1]
+ transmissionMode: TransmissionModeKind [0..1]
PanPricing
+ providerID: Integer [0..1]
+PanPricing
0..1
+PanPricingDetails
0..*
PanPricingDetail
+ alternateCostDelivered: Float [0..1]
+ alternateCostUnit: String [0..1]
+ currentTimeDate: DateTime [0..1]
+ generationPrice: Money [0..1]
+ generationPriceRatio: Float [0..1]
+ priceTierCount: Integer [0..1]
+ price: Money [0..1]
+ priceRatio: Float [0..1]
+ priceTier: Integer [0..1]
+ rateLabel: String [0..1]
+ registerTier: String [0..1]
+ unitOfMeasure: String [0..1]
+ priceTierLabel: String [0..1]
«Compound»
ControlledAppliance
+ isElectricVehicle: Boolean [0..1]
+ isExteriorLighting: Boolean [0..1]
+ isInteriorLighting: Boolean [0..1]
+ isGenerationSystem: Boolean [0..1]
+ isHvacCompressorOrFurnace: Boolean [0..1]
+ isIrrigationPump: Boolean [0..1]
+ isManagedCommercialIndustrialLoad: Boolean [0..1]
+ isPoolPumpSpaJacuzzi: Boolean [0..1]
+ isSimpleMiscLoad: Boolean [0..1]
+ isSmartAppliance: Boolean [0..1]
+ isStripAndBaseboardHeater: Boolean [0..1]
+ isWaterHeater: Boolean [0..1]
«enumeration»
TransmissionModeKind
normal
anonymous
both
IEC 326/13

Figure 49 – Class diagram Metering::MeteringEventsAndControls

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

Figure 50 shows class diagram MeteringMultipliers.

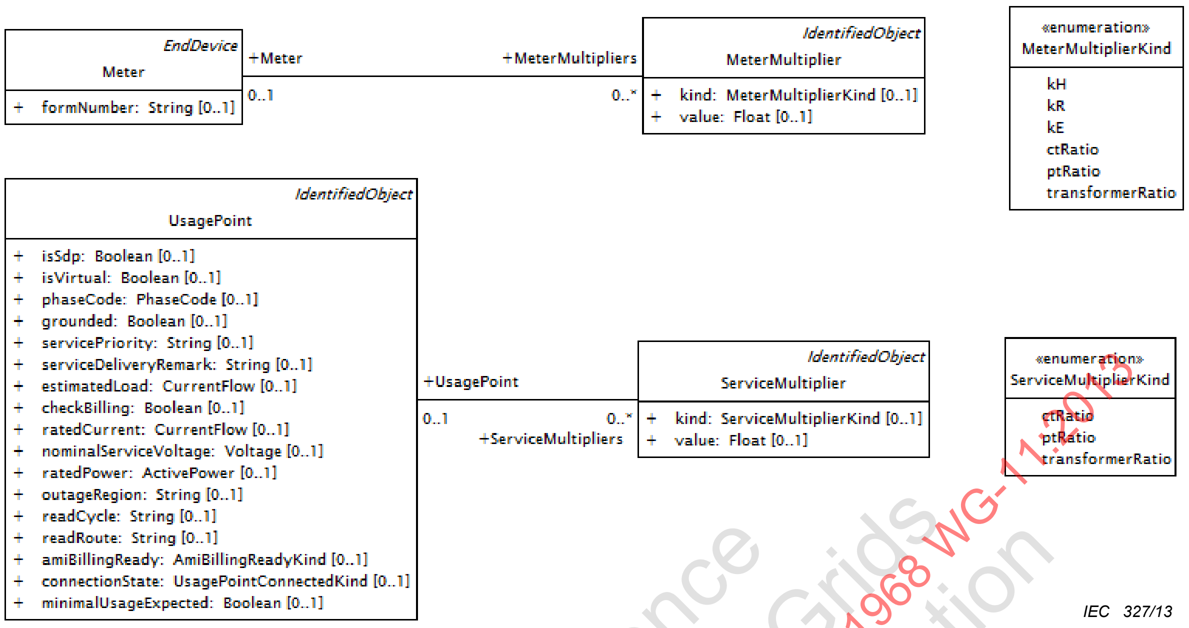

EndDevice
Meter
+ formNumber: String [0..1]
+Meter
0..1
+MeterMultipliers
0..*
IdentifiedObject
MeterMultiplier
+ kind: MeterMultiplierKind [0..1]
+ value: Float [0..1]
«enumeration»
MeterMultiplierKind
kH
kR
kE
ctRatio
ptRatio
transformerRatio
IdentifiedObject
UsagePoint
+ isSdp: Boolean [0..1]
+ isVirtual: Boolean [0..1]
+ phaseCode: PhaseCode [0..1]
+ grounded: Boolean [0..1]
+ servicePriority: String [0..1]
+ serviceDeliveryRemark: String [0..1]
+ estimatedLoad: CurrentFlow [0..1]
+ checkBilling: Boolean [0..1]
+ ratedCurrent: CurrentFlow [0..1]
+ nominalServiceVoltage: Voltage [0..1]
+ ratedPower: ActivePower [0..1]
+ outageRegion: String [0..1]
+ readCycle: String [0..1]
+ readRoute: String [0..1]
+ amiBillingReady: AmiBillingReadyKind [0..1]
+ connectionState: UsagePointConnectedKind [0..1]
+ minimalUsageExpected: Boolean [0..1]
+UsagePoint
0..1
0..*
+ServiceMultipliers
IdentifiedObject
ServiceMultiplier
+ kind: ServiceMultiplierKind [0..1]
+ value: Float [0..1]
«enumeration»
ServiceMultiplierKind
ctRatio
ptRatio
transformerRatio
IEC 327/13

Figure 50 – Class diagram Metering::MeteringMultipliers

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

Figure 51 shows class diagram MeteringTypes.

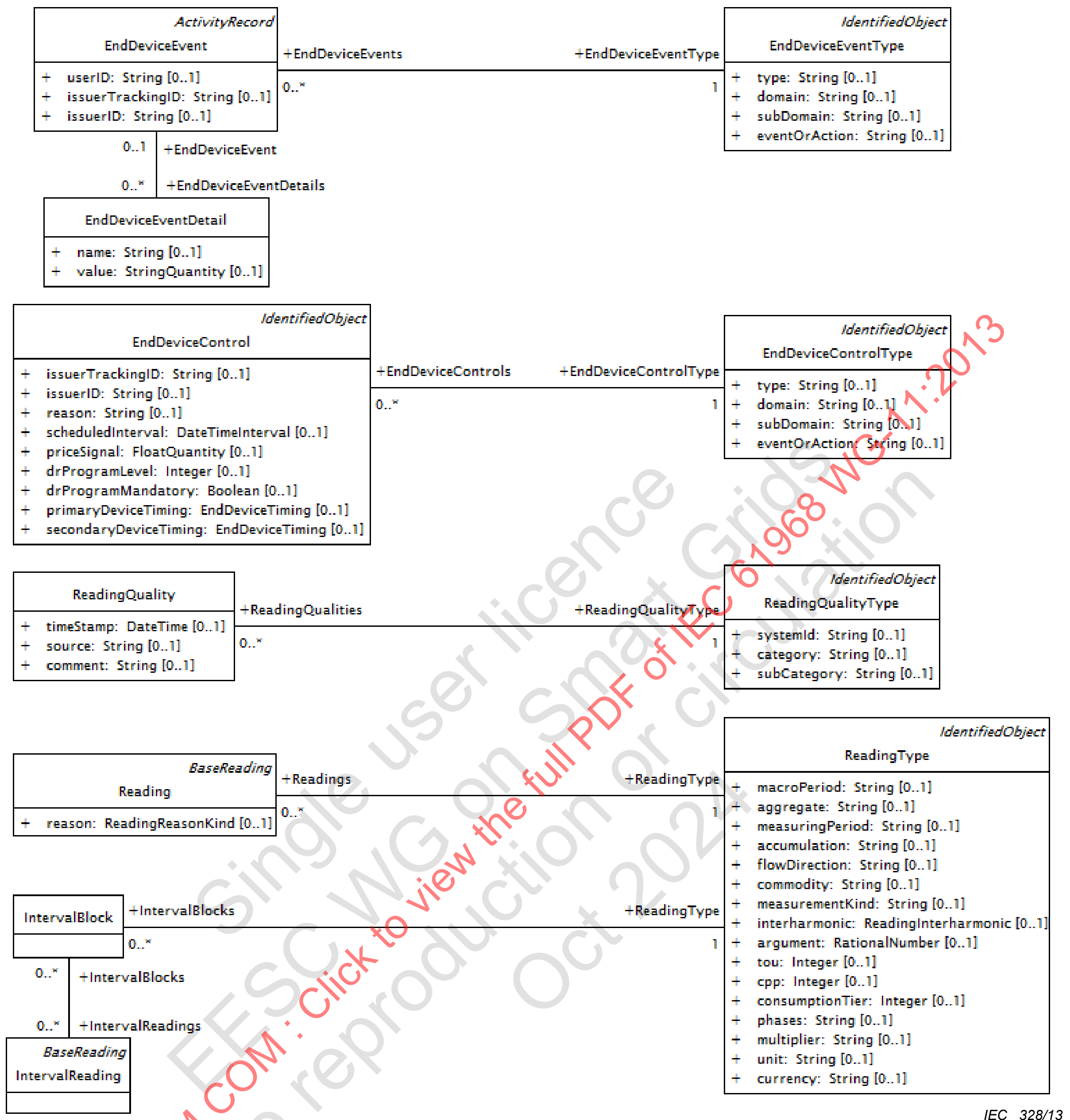

ActivityRecord
EndDeviceEvent
+ userID: String [0..1]
+ issuerTrackingID: String [0..1]
+ issuerID: String [0..1]
+EndDeviceEvents
0..*
+EndDeviceEventType
1
IdentifiedObject
EndDeviceEventType
+ type: String [0..1]
+ domain: String [0..1]
+ subDomain: String [0..1]
+ eventOrAction: String [0..1]
0..1
+EndDeviceEvent
0..*
+EndDeviceEventDetails
EndDeviceEventDetail
+ name: String [0..1]
+ value: StringQuantity [0..1]
IdentifiedObject
EndDeviceControl
+ issuerTrackingID: String [0..1]
+ issuerID: String [0..1]
+ reason: String [0..1]
+ scheduledInterval: DateTimeInterval [0..1]
+ priceSignal: FloatQuantity [0..1]
+ drProgramLevel: Integer [0..1]
+ drProgramMandatory: Boolean [0..1]
+ primaryDeviceTiming: EndDeviceTiming [0..1]
+ secondaryDeviceTiming: EndDeviceTiming [0..1]
+EndDeviceControls
0..*
+EndDeviceControlType
1
IdentifiedObject
EndDeviceControlType
+ type: String [0..1]
+ domain: String [0..1]
+ subDomain: String [0..1]
+ eventOrAction: String [0..1]
ReadingQuality
+ timeStamp: DateTime [0..1]
+ source: String [0..1]
+ comment: String [0..1]
+ReadingQualities
0..*
+ReadingQualityType
1
IdentifiedObject
ReadingQualityType
+ systemId: String [0..1]
+ category: String [0..1]
+ subCategory: String [0..1]
BaseReading
Reading
+ reason: ReadingReasonKind [0..1]
+Readings
0..*
+ReadingType
1
IdentifiedObject
ReadingType
+ macroPeriod: String [0..1]
+ aggregate: String [0..1]
+ measuringPeriod: String [0..1]
+ accumulation: String [0..1]
+ flowDirection: String [0..1]
+ commodity: String [0..1]
+ measurementKind: String [0..1]
+ interharmonic: ReadingInterharmonic [0..1]
+ argument: RationalNumber [0..1]
+ tou: Integer [0..1]
+ cpp: Integer [0..1]
+ consumptionTier: Integer [0..1]
+ phases: String [0..1]
+ multiplier: String [0..1]
+ unit: String [0..1]
+ currency: String [0..1]
IntervalBlock
+IntervalBlocks
0..*
+ReadingType
1
0..*
+IntervalBlocks
0..*
+IntervalReadings
BaseReading
IntervalReading
IEC 328/13

Figure 51 – Class diagram Metering::MeteringTypes

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

6.8.2 AmiBillingReadyKind enumeration

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

Table 125 shows all literals of AmiBillingReadyKind.

Table 125 – Literals of Metering::AmiBillingReadyKind

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

6.8.3 ComDirectionKind enumeration

Kind of communication direction.

Table 126 shows all literals of ComDirectionKind.

Table 126 – Literals of Metering::ComDirectionKind

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

6.8.4 ComTechnologyKind enumeration

Kind of communication technology.

Table 127 shows all literals of ComTechnologyKind.

Table 127 – Literals of Metering::ComTechnologyKind

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

6.8.5 EndDeviceFunctionKind enumeration

Kind of end device function.

Table 128 shows all literals of EndDeviceFunctionKind.

Table 128 – Literals of Metering::EndDeviceFunctionKind

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

6.8.6 MeterMultiplierKind enumeration

Kind of meter multiplier.

Table 129 shows all literals of MeterMultiplierKind.

Table 129 – Literals of Metering::MeterMultiplierKind

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

6.8.7 RandomisationKind enumeration

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

Table 130 shows all literals of RandomisationKind.

Table 130 – Literals of Metering::RandomisationKind

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

6.8.8 ReadingReasonKind enumeration

Reason for the reading being taken.

Table 131 shows all literals of ReadingReasonKind.

Table 131 – Literals of Metering::ReadingReasonKind

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

6.8.9 ServiceMultiplierKind enumeration

Kind of service multiplier.

Table 132 shows all literals of ServiceMultiplierKind.

Table 132 – Literals of Metering::ServiceMultiplierKind

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

6.8.10 TransmissionModeKind enumeration

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

Table 133 shows all literals of TransmissionModeKind.

Table 133 – Literals of Metering::TransmissionModeKind

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

6.8.11 UsagePointConnectedKind enumeration

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

Table 134 shows all literals of UsagePointConnectedKind.

Table 134 – Literals of Metering::UsagePointConnectedKind

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

6.8.12 ControlledAppliance compound

Appliance controlled with a PAN device control.

Table 135 shows all attributes of ControlledAppliance.

Table 135 – Attributes of Metering::ControlledAppliance

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

6.8.13 EndDeviceCapability compound

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

Table 136 shows all attributes of EndDeviceCapability.

Table 136 – Attributes of Metering::EndDeviceCapability

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

6.8.14 EndDeviceTiming compound

Timing for the control actions of end devices.

Table 137 shows all attributes of EndDeviceTiming.

Table 137 – Attributes of Metering::EndDeviceTiming

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

6.8.15 RationalNumber compound

Rational number = 'numerator' / 'denominator'.

Table 138 shows all attributes of RationalNumber.

Table 138 – Attributes of Metering::RationalNumber

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

6.8.16 ReadingInterharmonic compound

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

Table 139 shows all attributes of ReadingInterharmonic.

Table 139 – Attributes of Metering::ReadingInterharmonic

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

6.8.17 BaseReading

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

Table 140 shows all attributes of BaseReading.

Table 140 – Attributes of Metering::BaseReading

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

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

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

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

6.8.18 Channel

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

Table 142 shows all attributes of Channel.

Table 142 – Attributes of Metering::Channel

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

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

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

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

6.8.19 ComFunction

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

Table 144 shows all attributes of ComFunction.

Table 144 – Attributes of Metering::ComFunction

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

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

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

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

6.8.20 ComModule

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

Table 146 shows all attributes of ComModule.

Table 146 – Attributes of Metering::ComModule

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

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

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

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

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

6.8.21 DemandResponseProgram

Demand response program.

Table 148 shows all attributes of DemandResponseProgram.

Table 148 – Attributes of Metering::DemandResponseProgram

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

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

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

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

6.8.22 EndDevice

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

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

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

Table 150 shows all attributes of EndDevice.

Table 150 – Attributes of Metering::EndDevice

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

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

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

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

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

6.8.23 EndDeviceAction root class

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

Table 152 shows all attributes of EndDeviceAction.

Table 152 – Attributes of Metering::EndDeviceAction

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

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

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

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

6.8.24 EndDeviceControl

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

Table 154 shows all attributes of EndDeviceControl.

Table 154 – Attributes of Metering::EndDeviceControl

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

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

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

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

6.8.25 EndDeviceControlType

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

Table 156 shows all attributes of EndDeviceControlType.

Table 156 – Attributes of Metering::EndDeviceControlType

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

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

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

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

6.8.26 EndDeviceEvent

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

Table 158 shows all attributes of EndDeviceEvent.

Table 158 – Attributes of Metering::EndDeviceEvent

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

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

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

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

6.8.27 EndDeviceEventDetail root class

Name-value pair, specific to end device events.

Table 160 shows all attributes of EndDeviceEventDetail.

Table 160 – Attributes of Metering::EndDeviceEventDetail

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

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

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

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

6.8.28 EndDeviceEventType

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

Table 162 shows all attributes of EndDeviceEventType.

Table 162 – Attributes of Metering::EndDeviceEventType

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

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

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

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

6.8.29 EndDeviceFunction

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

Table 164 shows all attributes of EndDeviceFunction.

Table 164 – Attributes of Metering::EndDeviceFunction

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

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

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

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

6.8.30 EndDeviceGroup

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

Table 166 shows all attributes of EndDeviceGroup.

Table 166 – Attributes of Metering::EndDeviceGroup

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

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

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

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

6.8.31 EndDeviceInfo

End device data.

Table 168 shows all attributes of EndDeviceInfo.

Table 168 – Attributes of Metering::EndDeviceInfo

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

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

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

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

6.8.32 IntervalBlock root class

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

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

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

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

6.8.33 IntervalReading

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

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

Table 171 shows all attributes of IntervalReading.

Table 171 – Attributes of Metering::IntervalReading

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

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

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

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

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

6.8.34 Meter

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

Table 173 shows all attributes of Meter.

Table 173 – Attributes of Metering::Meter

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

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

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

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

6.8.35 MeterMultiplier

Multiplier applied at the meter.

Table 175 shows all attributes of MeterMultiplier.

Table 175 – Attributes of Metering::MeterMultiplier

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

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

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

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

6.8.36 MeterReading

Set of values obtained from the meter.

Table 177 shows all attributes of MeterReading.

Table 177 – Attributes of Metering::MeterReading

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

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

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

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

6.8.37 MeterServiceWork

Work involving meters.

Table 179 shows all attributes of MeterServiceWork.

Table 179 – Attributes of Metering::MeterServiceWork

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

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

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

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

6.8.38 MetrologyRequirement

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

Table 181 shows all attributes of MetrologyRequirement.

Table 181 – Attributes of Metering::MetrologyRequirement

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

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

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

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

6.8.39 PanDemandResponse

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

Table 183 shows all attributes of PanDemandResponse.

Table 183 – Attributes of Metering::PanDemandResponse

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

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

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

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

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

6.8.40 PanDisplay

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

Table 185 shows all attributes of PanDisplay.

Table 185 – Attributes of Metering::PanDisplay

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

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

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

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

6.8.41 PanPricing

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

Table 187 shows all attributes of PanPricing.

Table 187 – Attributes of Metering::PanPricing

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

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

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

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

6.8.42 PanPricingDetail root class

Detail for a single price command/action.

Table 189 shows all attributes of PanPricingDetail.

Table 189 – Attributes of Metering::PanPricingDetail

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