This document summarizes the use of a reduced-order CIM[[1]](#footnote-1) to support feeder modeling for the volt-var application in Release Cycle 1 (RC1). The full CIM includes over 1100 tables in SQL, each one corresponding to a UML class, enumeration or datatype. In RC1, we’re using approximately 100 such entities, mapped onto 100+ tables in SQL. Later versions of GridAPPS-D will use a triple-store or graph database, both of which appear to be better suited for CIM.

The CIM subset described here is based on the profile adopted for the most recent distribution CIM interoperability test, which was held in 2011 at EDF. For GridAPPS-D, we have updated that profile for compatibility with the most recent CIM base standard.

Table of Contents

[Class Diagrams for the Profile 1](#_Toc479167203)

[Typical Queries 13](#_Toc479167204)

[Object Diagrams for Queries 15](#_Toc479167205)

[Metering Relationship to Loads in the CIM 26](#_Toc479167206)

[CIM Enhancements for RC2 30](#_Toc479167207)

[CIM Profile in CIMTool 30](#_Toc479167208)

[Creating Data Definition Language (DDL) for MySQL 31](#_Toc479167209)

# Class Diagrams for the Profile

Figure 1 through Figure 11 present the UML class diagrams generated from Enterprise Architect[[2]](#footnote-2). These diagrams provide an essential roadmap for understanding:

1. How to ingest CIM XML from various sources into the database
2. How to generate native GridLAB-D input files from the database

For those unfamiliar with UML class diagrams:

1. Lines with an arrowhead indicate class inheritance. For example, in Figure 1, ACLineSegment inherits from Conductor, ConductingEquipment, Equipment and then PowerSystemResource. ACLineSegment inherits all attributes and associations from its ancestors (e.g. length), in addition to its own attributes and ancestors.
2. Lines with a diamond indicate composition. For example, in Figure 1, ConnectivityNodes make up a TopologicalNode, and then TopologicalNodes make up a TopologicalIsland.
3. Lines without a terminating symbol are associations. For example, in Figure 1, ACLineSegment has (through inheritance) a BaseVoltage, Location and EquipmentContainer.
4. Italicized names at the top of each class indicate the ancestor (aka superclass), in cases where the ancestor does not appear on the diagram. For example, in Figure 1, PowerSystemResource inherits from IdentifiedObject.

Please see *OSPRREYS\_RC1.eap[[3]](#footnote-3)* in the repository[[4]](#footnote-4) on GitHub for the latest updates. The EnterpriseArchitect file includes a description of each class, attribute and association. It can also generate HTML documentation of the CIM, with more detail than provided here.

The diagrammed UML associations have a role and cardinality at each end, source and target. In practice, only one end of each association is profiled and implemented in SQL. In some cases, the figure captions indicate which end, but see the CIM profile for specific definitions, as described in the object diagram section.

Nearly every CIM class inherits from IdentifiedObject, from which we use two attributes:

1. mRID is the “master identifier” that must be unique and persistent among all instances. It’s often used as the RDF resource identifier, and is often a GUID.
2. Name is a human-readable identifier that need not be unique.

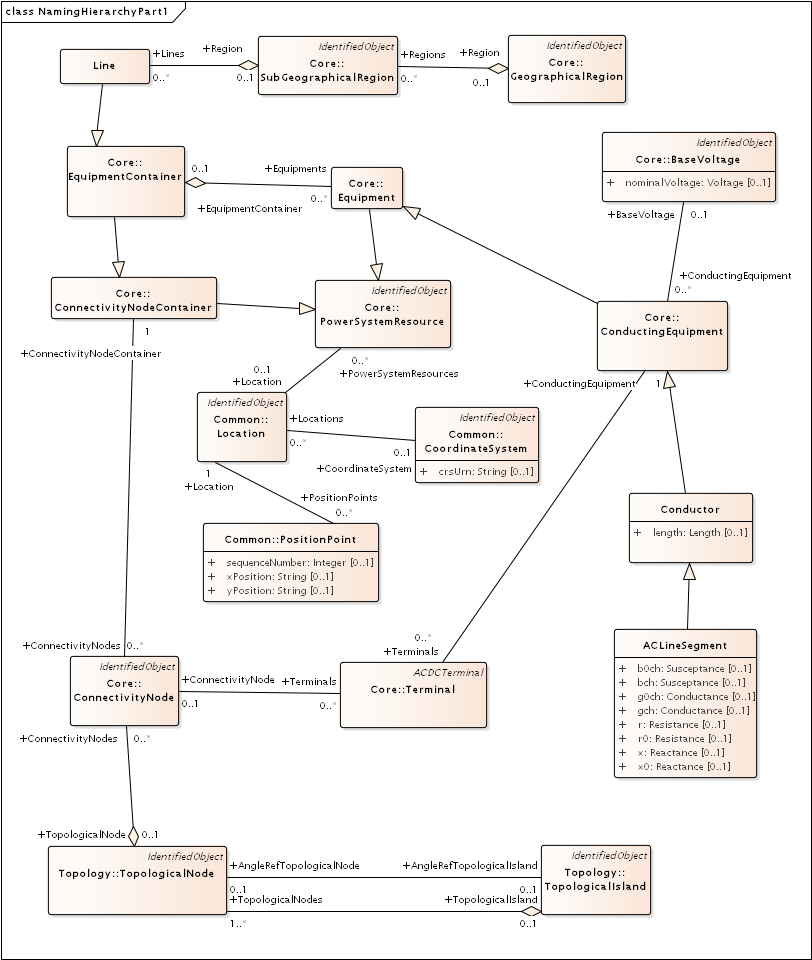


Figure 1: Placement of ACLineSegment into a Line (aka Feeder). In GridAPPS-D, the Line is the EquipmentContainer for all power system components and the ConnectivityNodeContainer for all nodes. It also corresponds to one TopologicalIsland. It’s part of a SubGeographicalRegion and GeographicalRegion for proper context with other CIM models. For visualization, ACLineSegment can be drawn from a sequence of PositionPoints associated via Location. The Terminals are free-standing; two of them will “reverse-associate” to the ACLineSegment as ConductingEquipment, and each terminal also has one ConnectivityNode. In RC1, we have a one-to-one association between ConnectityNode and TopologicalNode. The AngleRefTopologicalNode association can be used to identify the swing bus for GridLAB-D. Otherwise, we’re only using the topology classes to facilitate state variables, as described in Figure 11. The Terminal:phases attribute is not used; instead, phases will be defined in the ConductingEquipment instances. The associated BaseVoltage:nominalVoltage attribute is important for many of the classes that don’t have their own rated voltage attributes, for example, EnergyConsumer.

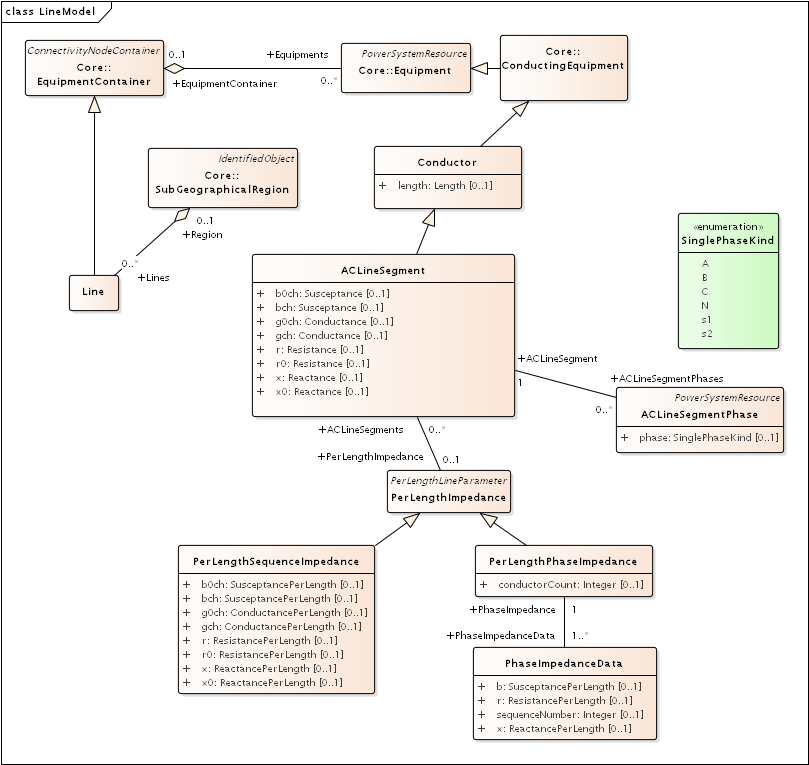


Figure 2: There are four different ways to specify ACLineSegment impedances. In all cases, Conductor:length is required. The first way is to specify the individual ACLineSegment attributes, which are sequence impedances and admittances, leaving PerLengthImpedance null. The second way is to specify the same attributes on an associated PerLengthSequenceImpedance, in which case the ACLineSegment attributes should be null. The third way is to associate a PerLengthPhaseImpedance, leaving the ACLineSegment attributes null. Only conductorCount from 1 to 3 is supported, and there will be 1, 3 or 6 reverse-associated PhaseImpedanceData instances that define the lower triangle of the Z and Y matrices per unit length. The sequenceNumber goes from 1 to N+N\*(N-1)/2 in column order. The fourth way to specify impedance is by wire/cable and spacing data, as described with Figure 10. If there are ACLineSegmentPhase instances reverse-associated to the ACLineSegment, then per-phase modeling applies. There are several use cases for ACLineSegmentPhase: 1) single-phase or two-phase primary, 2) low-voltage secondary using phases s1 and s2, 3) associated wire data where the neutral exists, 4) associated wire data where the phase wires are different. It is the application’s responsibility to propagate phasing through terminals to other components, and to identify any miswiring.

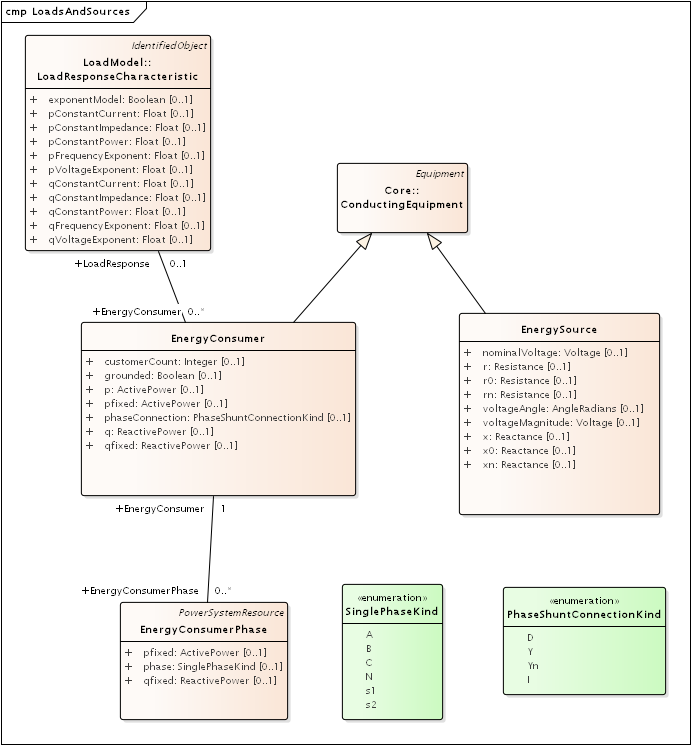


Figure 3: The EnergySource is balanced three-phase, representing a transmission system source (this is probably not the way we’ll model distributed generation in future versions). The EnergyConsumer is a ZIP load, possibly unbalanced, with an associated LoadResponse instance defining the ZIP coefficients. For three-phase delta loads, the phaseConnection is D and the three reverse-associated EnergyConsumerPhase instances will have phase=A for the AB load, phase=B for the BC load and phase=C for the AC load. A three-phase wye load may have either Y or Yn for the phaseConnection. Single-phase and two-phase loads, including secondary loads, should have phaseConnection=I (for individual).

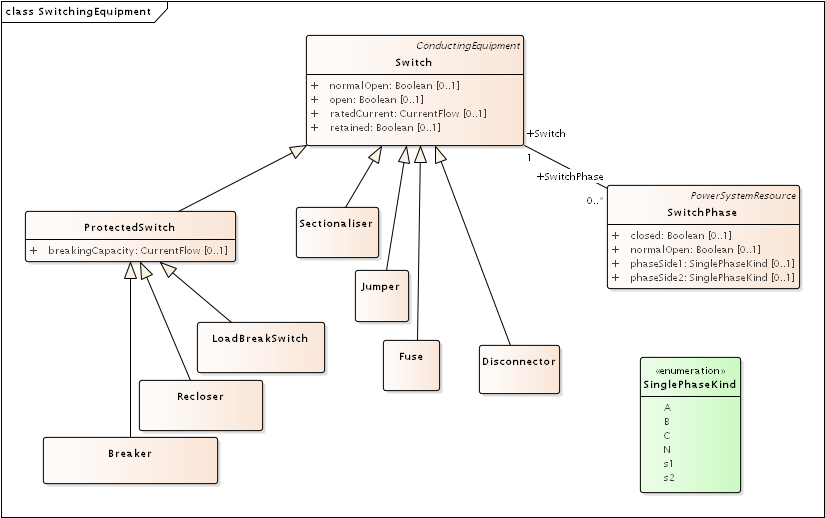


Figure 4: There are seven different kinds of Switch supported in the CIM, and all of them have zero impedance. They would all behave the same in power flow analysis, and all would require many more attributes than are defined in CIM to support protection analysis. The use cases for SwitchPhase include 1) single-phase, two-phase and secondary switches, 2) one or two conductors open in a three-phase switch or 3) transpositions, in which case phaseSide1 and phaseSide2 would be different.

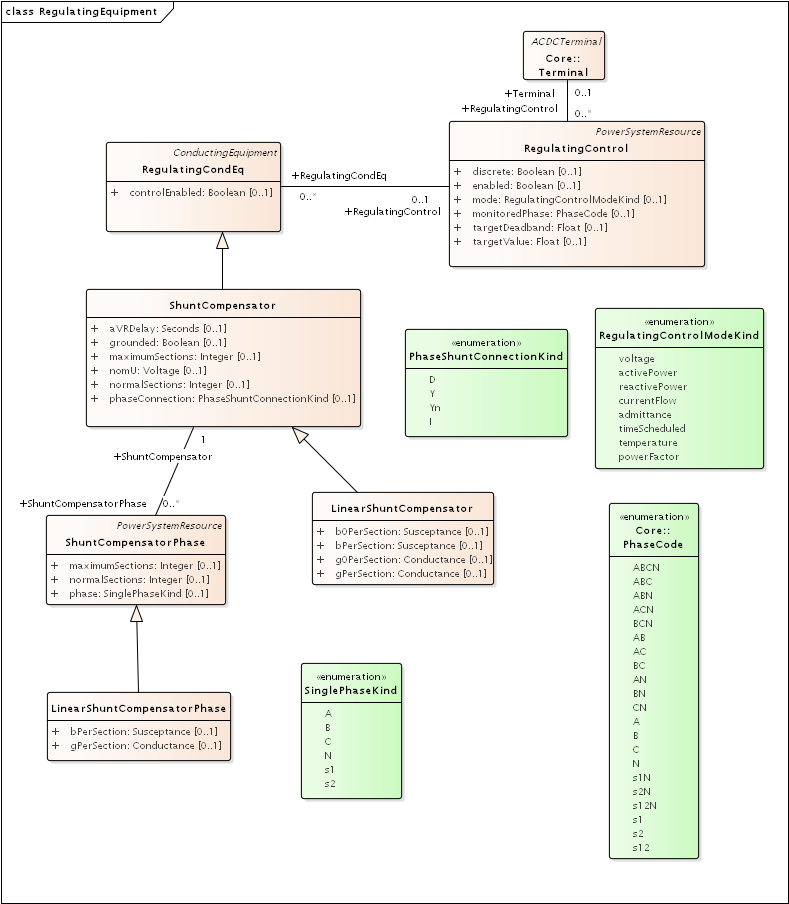


Figure 5: On the left, LinearShuntCompensator and LinearShuntCompensatorPhase define capacitor banks, in a way very similar to EnergyConsumer in Figure 3. The kVAR ratings must be converted to susceptance based on the nominal voltage, nomU. Note that aVRDelay is really a capacitor control parameter, to be used in conjunction with RegulatingControl on the right-hand side. The RegulatingControl associates to the controlled capacitor bank via RegulatingCondEq, and to the monitored location via Terminal. There is no support for a PT or CT ratio, so targetDeadband and targetValue have to be in primary volts, amps, vars, etc.

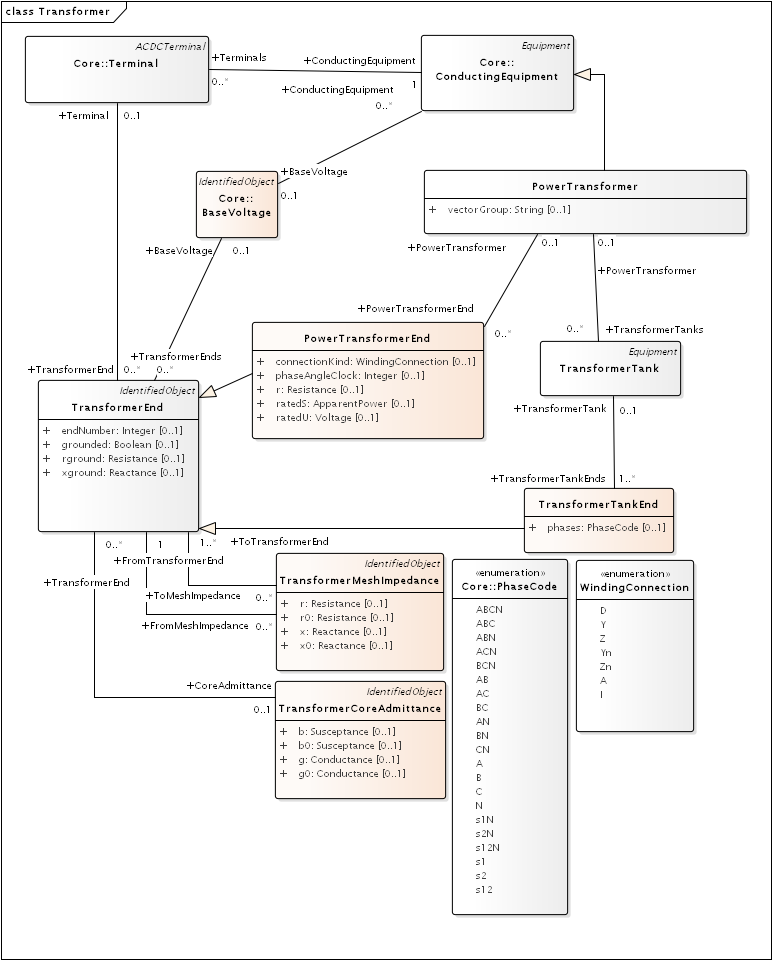


Figure 6: PowerTransformers may be modeled with or without tanks, and in both cases vectorGroup should be specified according to IEC transformer standards (e.g. Dy1 for many substation transformers). The case without tanks is most suitable for balanced three-phase transformers that won’t reference catalog data; any other case should use tank-level modeling. In the tankless case, each winding will have a PowerTransformerEnd that associates to both a Terminal and a BaseVoltage, and the parent PowerTransformer. The impedance and admittance parameters are defined by reverse-associated TransformerMeshImpedance between each pair of windings, and a reverse-associated TransformerCoreAdmittance for one winding. The units for these are ohms and siemens based on the winding voltage, rather than per-unit. WindingConnection is similar to PhaseShuntConnectionKind, adding Z and Zn for zig-zag connections and A for autotranformers. If the transformer is unbalanced in any way, then TransformerTankEnd is used instead of PowerTransformerEnd, and then one or more TransformerTanks may be used in the parent PowerTransformer. Some of the use cases are 1) center-tapped secondary, 2) open-delta and 3) EHV transformer banks. Tank-level modeling is also required is using catalog data, as described with Figure 9.

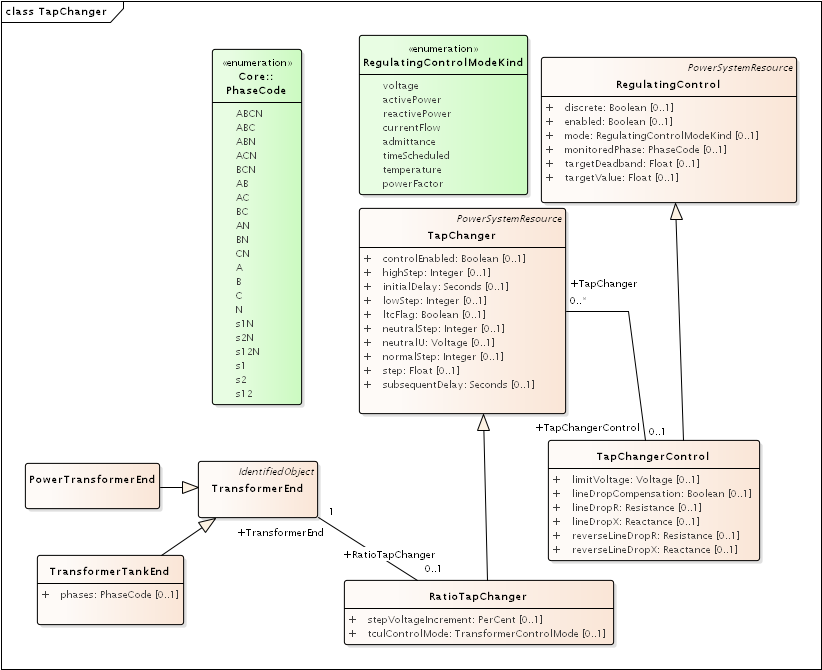


Figure 7: A RatioTapChanger can represent a transformer tap changer on the associated TransformerEnd. The RatioTapChanger has some parameters defined in a direct-associated TapChangerControl, which inherits from RegulatingControl some of the same attributes used in capacitor controls (Figure 5). Therefore, a line voltage regulator in CIM includes a PowerTransformer, a RatioTapChanger, and a TapChangerControl. The CT and PT parameters of a voltage regulator can only be described via the AssetInfo mechanism, described with Figure 8.

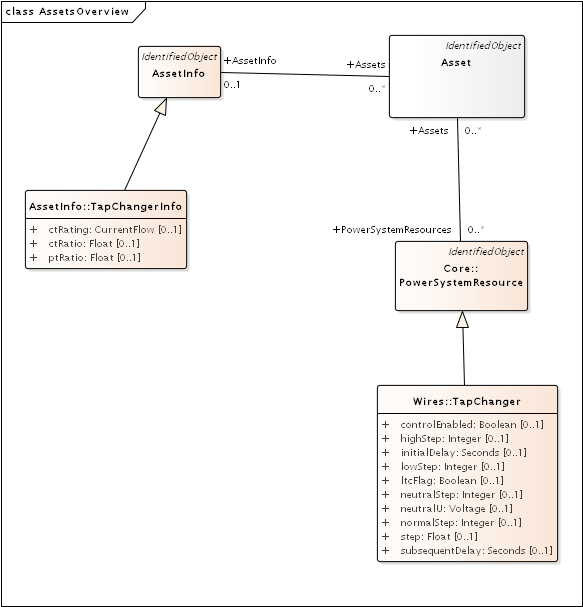


Figure 8: Many distribution software packages use the concept of catalog data, aka library data, especially for lines and transformers. We use the Asset and AssetInfo packages to implement this in CIM. Here, the TapChangerInfo class includes the CT rating, CT ratio and PT ratio parameters needed for line drop compensator settings in voltage regulators. Catalog data is a one-to-many, and sometimes a many-to-many, relationship. For these lookups, we create an Asset instance that has one association to AssetInfo, and one-to-many associations to PowerSystemResources. In this case, many TapChangers can share the same TapChangerInfo data, which saves space and provides consistency.

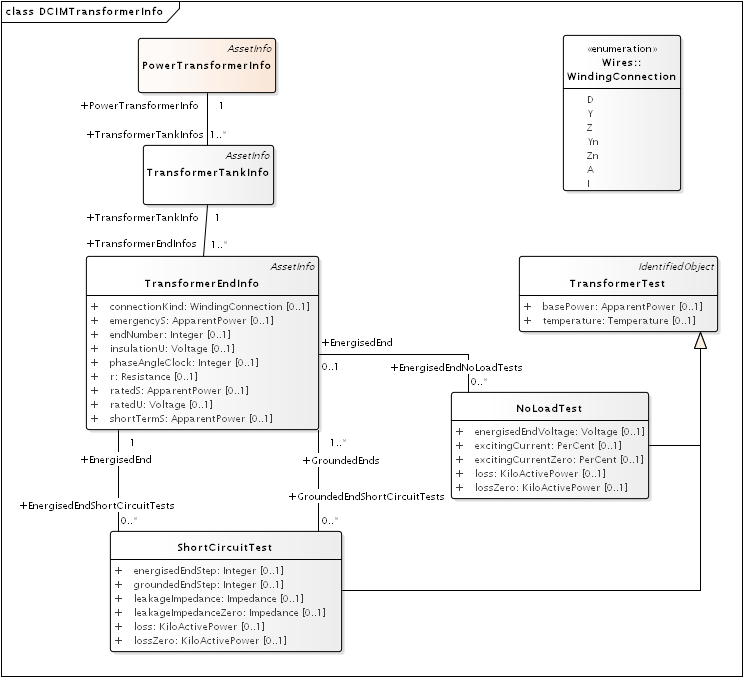


Figure 9: The catalog mechanism for transformers will associate a TransformerTank (Figure 6) with TransformerTankInfo (here), via the one-to-many mechanism described in Figure 8. The PowerTransformerInfo collects TransformerTankInfo by reverse association, but it does not link with PowerTransformer. In other words, the physical tanks are cataloged because transformer testing is done on tanks. One possible use for PowerTransformerInfo is to help organize the catalog. It’s important that TransformerEndInfo:endNumber (here) properly match the TransformerEnd:endNumber (Figure 6). The shunt admittances are defined by NoLoadTest on a winding / end, usually just one such test. The impedances are defined by a set of ShortCircuitTests; one winding / end will be energized, and one or more of the others will be grounded in these tests.

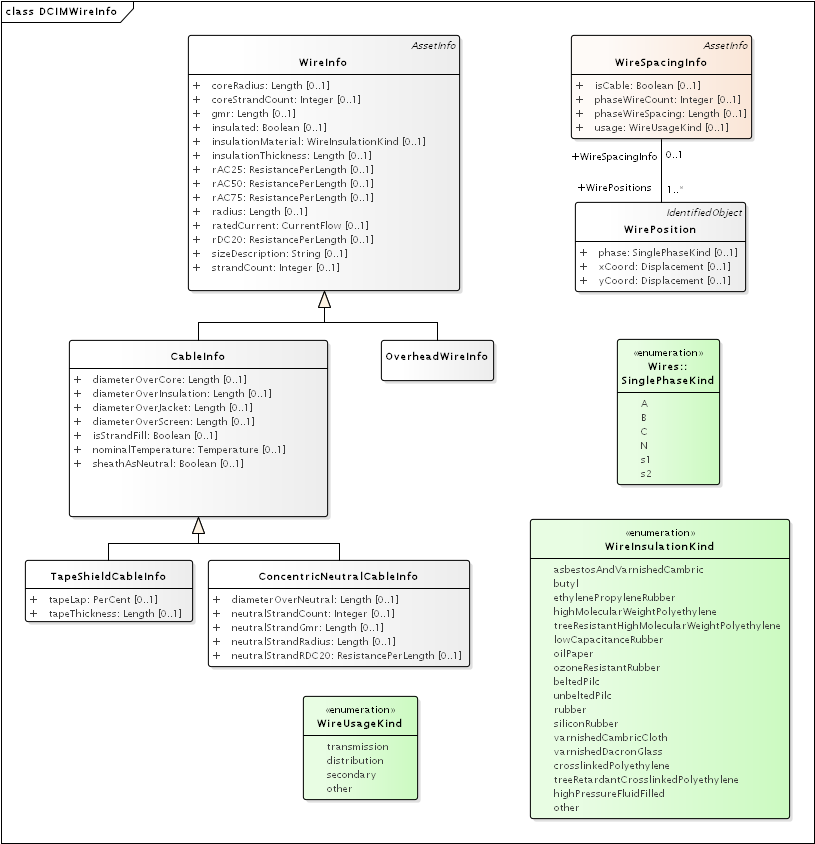


Figure 10: The catalog / library mechanism for ACLineSegment will have a WireSpacingInfo associated as in Figure 9. This will indicate whether the line is overhead or underground. phaseWireCount and phaseWireSpacing define optional bundling, so these will be 1 and 0 for distribution. The number of phase and neutral conductors is actually defined by the number of reverse-associated WirePosition instances. For example, a three-phase line with neutral would have four of them, with phase = A, B, C and N. On the right-hand side, concrete classes OverheadWireInfo, TapeShieldCableInfo and ConcentricNeutralCableInfo may be associated (as in Figure 9) to either ACLineSegment or ACLineSegmentPhase. The association to ACLineSegment only applies for three-conductor, three-phase lines all using the same wire data, or to supply just the ratedCurrent attribute. All other use cases would associate to ACLineSegmentPhase. It’s the application’s responsibility to calculate impedances from this data. In particular, soil resistivity and dielectric constants are not included in the CIM. Typical dielectric constant values might be defined for each WireInsulationKind.

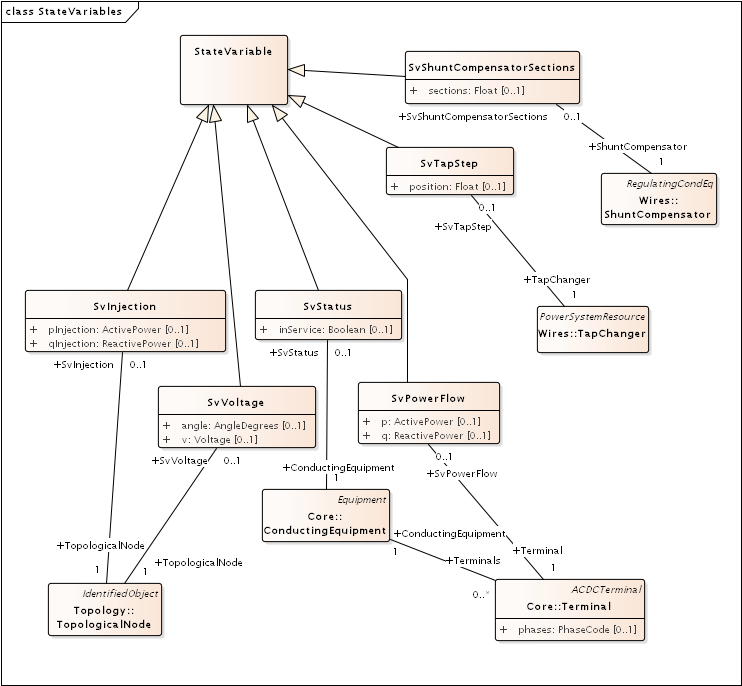


Figure 11: The CIM state variables package might be used to mimic sensor locations and values on the distribution system. Voltages are measured on TopologicalNodes, power flows are measured at Terminals, step positions are measured on TapChangers, status is measured on ConductingEquipment, and on/off state is measured on ShuntCompensators. The “injections” have been included here, but there may not be a use case for them in distribution. On the other hand, we would need an SvCurrent, which was probably not included in the CIM because of its transmission system heritage. Attributes for sensor characteristics would also have to be added in future versions of GridAPPS-D.

# Typical Queries

These queries focus on requirements of the first volt-var application.

1. Capacitors (Figure 5, Figure 12, Figure 13, Figure 14)
   1. Create a list of capacitors with bus name (Connectivity Node in Figure 1), kVAR per phase, control mode, target value and target deadband
   2. For a selected capacitor, update the control mode, target value, and target deadband
2. Regulators (Figure 7, Figure 8, Figure 12, Figure 29)
   1. List all transformers that have a tap changer attached, along with their bus names and kVA sizes
   2. Given a transformer that has a tap changer attached, list or update initialDelay, step, subsequentDelay, mode, targetDeadband, targetValue, limitVoltage, lineDropCompensation, lineDropR, lineDropX, reverseLineDropR and reverseLineDropX
3. Transformers (Figure 6, Figure 9)
   1. Given a bus name or load (Figure 3), find the transformer serving it (Figure 16, Figure 19)
   2. Find the substation transformer, defined as the largest transformer (by kVA size and or highest voltage rating)
   3. List the transformer catalog (Figure 9, Figure 20) with name, highest ratedS, list of winding ratedU in descending order, vector group (https://en.wikipedia.org/wiki/Vector\_group used with connectionKind and phaseAngleClock), and percent impedance
   4. List the same information as in item c, but for transformers (Figure 6) and also retrieving their bus names. Note that a transformer can be defined in three ways
      1. Without tanks, for three-phase, multi-winding, balanced transformers (Figure 16 and Figure 17).
      2. With tanks along with TransformerTankInfo (Figure 9) from a catalog of “transformer codes”, which may describe balanced or unbalanced transformers. See Figure 19 and Figure 20.
      3. With tanks for unbalanced transformers, and TransformerTankInfo created on-the-fly. See Figure 19 and Figure 20.
   5. Given a transformer (Figure 6), update it to use a different catalog entry (TransformerTankInfo in Figure 9)
4. Lines (Figure 2, Figure 10, Figure 12)
   1. List the line and cable catalog entries that meet a minimum ratedCurrent and specific WireUsageKind. For cables, be able to specify tape shield vs. concentric neutral, the WireInsulationKind, and a minimum insulationThickness. (Figure 27)
   2. Given a line segment (Figure 2) update to use a different linecode (Figure 10, Figure 26)
   3. Given a bus name, list the ACLineSegments connected to the bus, along with the length, total r, total x, and phases used. There are four cases as noted in the caption of Figure 2, and see Figure 23 through Figure 26.
   4. Given a bus name, list the set of ACLineSegments (or PowerTransformers and Switches) completing a path from it back to the EnergySource (Figure 3). Normally, the applications have to build a graph structure in memory to do this, so it would be very helpful if a graph/semantic database can do this.
5. Voltage and other measurements (Figure 1, Figure 11)
   1. Given a bus, attach a voltage measurement point (SvVoltage, Figure 30)
   2. List all voltage measurement points and their buses, and for each bus, list the phases actually present
   3. For tap changer position (SvTapStep, Figure 31), attach and list measurements as in items a and b
   4. For capacitor switch status (SvShuntCompensatorSections, Figure 32), attach and list measurements as in items a and b
6. Loads (Figure 3, Figure 28)
   1. Given a bus name, list and total all of the loads connected by phase, showing the total p and q, and the composite ZIP coefficients
7. Switching (Figure 4, Figure 22)
   1. Given a bus name, trace back to the EnergySource and list the switches encountered, grouped by type (i.e. the leaf class in Figure 4). Also include the ratedCurrent, breakingCapacity if applicable, and open/close status. If SwitchPhase is used, show the phasing on each side and the open/close status of each phase.
   2. Given switch, toggle its open/close status.

# Object Diagrams for Queries

This section contains UML object diagrams for the purpose of illustrating how to perform typical queries and updates. For those unfamiliar with UML object diagrams:

1. Each object will be an instance of a class, and more than one instance of a class can appear on the diagram. For example, Figure 12 shows two ConnectivityNode instances, one for each end of a ConductingEquipment.
2. The object name (if specified and important) appears before the colon (:) above the line, while the UML class appears after the colon. Every object in CIM will have a unique ID, and a name (not necessarily unique), even if not shown here.
3. Some objects may be shown with run-time state below the line. These are attribute value assignments, drawn from those available in the UML class or one of the class ancestors. The object may have more attribute assignments, but only those directly relevant to the figure captions are shown in the diagrams of this section.
4. Object associations are shown with solid lines, role names, and multiplicities similar to the UML class diagrams. One important difference is that only one way of navigating a particular association will be defined in the profile. For example, the lower left corner of Figure 1 shows a two-way link between TopologicalNode and ConnectivityNode in the UML class diagram. However, Figure 12 shows that only one direction has been defined in the profile. Each ConnectivityNode has a direct reference to its corresponding TopologicalNode. In order to navigate the reverse direction from TopologicalNode to ConnectivityNode, some type of conditional query would be required. In other words, the object diagrams in this section indicate which associations can actually be used in GridAPPS-D.
5. In some cases, the multiplicities on the object diagrams are more restrictive than on the class diagrams, due to profiling. For example, Figure 12 reflects a one-to-one correspondence between ConnectivityNode and TopologicalNode in this profile.

The object diagrams are intended to help you break down the CIM queries into common sub-tasks. For example, query #1 works with capacitors. It’s always possible to select a capacitor (aka LinearShuntCompensator) by name. In order to find the capacitor at a bus, say “bus1” in Figure 12, one would retrieve all Terminals having a ConnectivityNode reference to “bus1”. Each of those Terminals will have a ConductingEquipment reference, and you want the Terminal(s) for which that reference is actually a LinearShuntCompensator. In this CIM profile, only leaf classes (e.g. LinearShuntCompensator) will be instantiated, never base classes like ConductingEquipment. There can be more than one capacitor at a bus, more than one load, more than one line, etc.



Figure 12: In order to traverse buses and components, begin with a ConnectivityNode (left). Collect all terminals referencing that ConnectivityNode; each Terminal will have one-to-one association with ConductingEquipment, of which there are many subclasses. In this example, the ConductingEquipment has a second terminal referencing the ConnectivityNode called bus2. There are applications for both Depth-First Search (DFS) and Bread-First Search (BFS) traversals. Note 1: the Terminals have names, but these are not useful. Some Terminal names have been shown above, just to illustrate there is no useful implication of sequencing or ordering. Note 2: in this version of GridAPPS-D, we have one-to-one association of TopologicalNode and ConnectivityNode, but all searches should visit ConnectivityNodes. Note 3: transformers are subclasses of ConductingEquipment, but we traverse connectivity via transformer ends (aka windings). This is illustrated later.

In order to find capacitors (or anything else) associated with a particular “feeder”, Figure 13 shows that you would query for objects having EquipmentContainer reference to the feeder’s Line object. In GridAPPS-D RC1, we only use Line for equipment container in CIM, and this would correspond to one entire GridLAB-D model. There is also a BaseVoltage reference that will have the system nominal voltage for the capacitor’s location. However, in order to work with equipment ratings you should use ratedS and ratedU attributes where they exist, particularly for capacitors and transformers. These attributes are often slightly different than the “system voltage”. Most of the attribute units in CIM are SI, with a few exceptions like percent and kW values on transformer test sheets (i.e. CIM represents the test sheet, not the equipment).



Figure 13: All conducting equipment lies within an EquipmentContainer, which in GridAPPS-D, will be a Line object named after the feeder. It also has reference to a BaseVoltage, which is typically one of the ANSI preferred system voltages. Power transformers are a little different, in that each winding (called “end” in CIM) has reference to a BaseVoltage. Note that equipment ratings come from the vendor, and in this case ratedU is slightly different from nominalVoltage. All conducting equipment has a Location, which contains XY coordinates (see Figure 1). The Location is useful for visualization, but is not essential for a power flow model.

Completing the discussion of capacitors, Figure 14 provides two examples for single-phase, and three-phase with local voltage control. As shunt elements, capacitors have only one Terminal instance. Loads and sources have one terminal, lines and switches have two terminals, and transformers have two or more terminals. Examples of all those are shown later. In Figure 14, the capacitor’s kVAR rating will be  based on its nameplate ratedU, not the system’s nominalVoltage.

Often, the question will arise “what phases exist at this bus?”. There is no phasing explicitly associated with a ConnectivityNode or Terminal in CIM. To answer this question, we’d have to query for all ConductingEquipment instances having Terminals connected to that bus, as in Figure 12. The types of ConductingEquipment that may have individual phases include LinearShuntCompensators (Figure 14), ACLineSegments, PowerTransformers (via TransformerEnds), EnergyConsumers, and descendants of Switch. If the ConductingEquipment has such individual phases, then add those phases to list of phases existing at the bus. If there are no individual phases, then ABC all exist at the bus. Note this doesn’t guarantee that all wiring to the bus is correct; for example, you could still have a three-phase load served by only a two-phase line, which would be a modeling error. In Figure 14, we’d find phase C at Bus611 and phases ABC at Bus675. Elsewhere in the model, there should be ACLineSegments, PowerTransformers or Switch descendants delivering phase C to Bus611, all three phases ABC to Bus675.



Figure 14: Capacitors are called LinearShuntCompensator in CIM. On the left, a 100 kVAR, 2400 V single-phase bank is shown on phase C at bus 611. bPerSection = 100e3 / 2400^2 [S], and the bPerSection on LinearShuntCompensatorPhase predominates; these values can differ among phases if there is more than one phase present. On the right, a balanced three-phase capacitor is shown at bus 675, rated 300 kVAR and 4160 V line-to-line. We know it’s balanced three phase from the absence of associated LinearShuntCompensatorPhase objects. bPerSection = 300e4 / 4160^2 [S]. This three-phase bank has a voltage controller attached with 2400 V setpoint and 240 V deadband, meaning the capacitor switches ON if the voltage drops below 2280 V and OFF if the voltage rises above 2520 V. These voltages have to be monitored line-to-neutral in CIM, with no VT ratio. In this case, the control monitors the same Terminal that the capacitor is connected to, but a different conducting equipment’s Terminal could be used. The control delay is called aVRDelay in CIM, and it’s an attribute of the LinearShuntCompensator instead of the RegulatingControl. It corresponds to “dwell time” in GridLAB-D.

Figure 15 through Figure 20 illustrate the transformer query tasks, plus Figure 29 for attached voltage regulators. The autotransformer example is rated 500/345/13.8 kV and 500/500/50 MVA, for a transmission system. The short circuit test values are ZHL=10%, ZHT=25% and ZLT=30%. The no-load test values are 0.05% exciting current and 0.025% no-load losses. These convert to r, x, g and b in SI units, from  and  , where Srated and Urated are based on the “from” winding (aka end). The same base quantities would be used to convert r, x, g and b back to per-unit or percent. The open wye – open delta impedances are already represented in percent or kW, from the test reports.



Figure 15: Autotransformer with delta tertiary winding acts like a wye-wye transformer with smaller delta tertiary. The vector group would be Yynd1 or Yyd1. For analyses other than power flow, it can be represented more accurately as the physical series (n1) – common (n2) connection, with a vector group Yand1. In either case, it’s a three-winding transformer.



Figure 16: A three-winding autotransformer is represented in CIM as a PowerTransformer with three PowerTransformerEnds, because it’s balanced and three-phase. The three Terminals have direct ConductingEquipment references to the PowerTransformer, so you can find it from bus1, busX or busY. However, each PowerTransformerEnd has a back-reference to the same Terminal, and it’s own reference to BaseVoltage (Figure 13); that’s how you link the matching buses and windings, which must have compatible voltages. Terminals have no sequence number, so the endNumber is important for correct linkage to catalog data as discussed later. By convention, ends with highest ratedU have the lowest endNumber, and endNumber establishes that end’s place in the vectorGroup.



Figure 17: Power transformer impedances correspond to the three-winding autotransformer example of Figure 15 and Figure 16. There are three instances of TransformerMeshImpedance connected pair-wise between the three windings / ends. The x and r values are in Ohms referred to the end with highest ratedU in that pair. There is just one TransformerCoreAdmittance, usually attached to the end with lowest ratedU, and the attribute values are Siemens referred to that end’s ratedU.



Figure 18: Open wye - open delta transformer banks are used to provide inexpensive three-phase service to loads, by using only two single-phase transformers. This is an unbalanced transformer, and as such it requires tank modeling in CIM. Physically, the two transformers would be in separate tanks. Note that Tank A is similar to the residential center-tapped secondary transformer, except the CIM phases would include s1 and s2 instead of A and B.



Figure 19: Unbalanced PowerTransformer instances comprise one or more TransformerTanks, which own the TransformerTankEnds. Through the ends, busHi collects phases ABN and busLo collects phases ABCN. Typically, phase C will also exist at busHi, but this transformer doesn’t require it. We still assign vectorGroup Yd1 to the supervising PowerTransformer, as this is the typical case. The modeler should determine that. By comparison to Figure 19, there is a possible ambiguity in how endA3 represents the polarity dot at the neutral end of Wdg A3. An earlier CIM proposal would have assigned phaseAngleClock = 6 on endA3, but the attribute was removed from TransformerTankEnd. It may not be possible to infer the correct winding polarities from the vectorGroup in all cases. There is a phaseAngleClock attribute on TransformerTankEndInfo, but that represents a shelf state of the tank, not necessarily connections in the field. Therefore, it may be necessary to propose the phaseAngleClock attribute for TransformerTankEnd.



Figure 20: This Asset catalog example defines the impedances for Tank B of the open wye – open delta bank. This is a 50 kVA, 7200 / 240 V single-phase transformer. It has 1% exciting current and 0.4 kW loss in the no-load test, plus 2.1% reactance and 0.5 kW loss in the short-circuit test. A multi-winding transformer could have more than one grounded end in a short-circuit test, but this is not common. The catalog data is linked with one or more TransformerTanks via the Asset instance, shown to the left. This Asset instance won’t exist without such links (i.e. the catalog data is actually used), so cardinalities are 1 for AssetInfo and 1..\* for PowerSystemResources. Furthermore, endNumber on the TransformerEndInfo has to match endNumber on the TransformerTankEnd instances associated to Tank B. Instead of catalog information, we could have used mesh impedance and core admittance as in Figure 17, but we’d have to convert the test sheets to SI units and we could not share data with other TransformerTank instances, both of which are inconvenient.

Figure 21 through Figure 27 illustrate the query tasks for ACLineSegments and Switches, which will define most of the circuit’s connectivity. The example sequence impedances were based on Z1 = 0.1 + j0.8 /mile and Z0 = 0.5 + j2.0  /mile. For distribution systems, use of the shared catalog data is more common, either pre-calculated matrix (Figure 25) or spacing and conductor (Figure 26 and Figure 27). In both cases, impedance calculation is outside the scope of CIM (e.g. GridLAB-D internally calculates line impedance from spacing and conductor data).



Figure 21: An ACLineSegment with two phases, A and C. If there are no ACLineSegmentPhase instances that associate to it, assume it’s a three-phase ACLineSegment. This adds phases AC to bus671 and bus684.



Figure 22: This 50-Amp load break switch connects phases AC between busLeft and busRight. Without associated SwitchPhase instances, it would be a three-phase switch. This switch also transposes the phases; A on side 1 connects with C on side 2, while C on side 1 connects with A on side 2. This is the only way of transposing phases in CIM. Note the ambiguity in side 1 and side 2, because Terminal.sequenceNumber was subsequently removed from the CIM. This needs to be addressed in a future version of the CIM. Also note that LoadBreakSwitch has the open attribute inherited from Switch, while SwitchPhase has the converse closed attribute. In order to open and close the switch, these attributes would be toggled appropriately. See Figure 4 for other types of switch.



Figure 23: This is a balanced three-phase ACLineSegment between bus632 and bus671, 2000 feet or 609.6 m long. Sequence impedances are specified in ohms, as attributes on the ACLineSegment. This is a typical pattern for transmission lines, but not distribution lines.



Figure 24: The impedances from Figure 23 were divided by 609.6 m, to obtain ohms per meter for seqCat1. Utilities often call this a “line code”, and other ACLineSegment instances can share the same PerLengthImpedance. A model imported into the CIM could have many line codes, not all of them used in that particular model. However, those line codes should be available for updates by reassigning PerLengthImpedance.



Figure 25: This is a two-phase line segment from bus671 to bus684 using a line code, which has been specified using a 2x2 symmetric matrix of phase impedances per meter, instead of sequence impedances per meter. This is more common for distribution than either Figure 23 or Figure 24. It’s distinguished from Figure 24 by the fact that PerLengthImpedance references an instance of PerLengthPhaseImpedance, not PerLengthSequenceImpedance. The conductorCount attribute tells us it’s a 2x2 matrix, which will have two unique diagonal elements and one distinct off-diagonal element. The elements are provided in three PhaseImpedanceData instances, which are named here for clarity as Z11, Z12 and Z22. However, the sequenceNumber is most significant, as the elements must be numbered in lower triangular form. Finally, note that Z11 and Z22 are slightly different. The matrix row numbers must correspond to the phases present in ABC order. CIM doesn’t provide a way of transposing matrix row assignments, so in order to swap phases A and C, we’d have to create a second instance of PerLengthPhaseImpedance, with Z11 and Z22 swapped. The GridAPPS-D CIM importer will create these automatically, which expands the set of line codes. As presented here, mtx604 can apply to phasing AB, BC or AC.



Figure 26: The two-phase ACLineSegment impedance defined by sharing wire and spacing data from a catalog. Each ACLineSegmentPhase links to an OverheadWireInfo instance via the Asset instance. If the neutral (N) is present, we have to specify its wire information for a correct impedance calculation. In this case, ACN all use the same wire type, but they can be different, especially for the neutral. Similarly, the WireSpacingInfo associates to the ACLineSegment itself via a separate Asset instance. These Asset instances only exist when the catalog data is used, so cardinalities are 1 for AssetInfo and 1..\* for PowerSystemResources.



Figure 27: The upper five instances define catalog attributes for Figure 26. The WirePosition xCoord and yCoord units are meters, not feet, and they include explicit phase assignments to match ACLineSegmentPhase. This removes any ambiguity, but it’s still necessary to create copies for phase transposition. The phaseWireSpacing and phaseWireCount attributes are for sub-conductor bundling on EHV and UHV transmission lines; bundling is not used on distribution. The number of WirePositions that reference spc505acn determine how many wires need to be assigned, and the phase attributes in those WirePosition instances determine how many phases and neutrals there are. Eliminating the neutral, this would produce a 2x2 phase impedance matrix. Although the pattern appears general enough to support multiple neutrals and transmission overbuild, the CIM doesn’t actually have the required phasing codes. When isCable is true, the WirePosition yCoord values would be negative for underground depth. To find overhead wires of a certain size or ampacity, we can put query conditions on the ratedCurrent attribute. To find underground conductors, we query the ConcentricNeutralCableInfo or TapeShieldCableInfo instead of OverheadWireInfo. All three inherit the ratedCurrent attribute from WireInfo. Cables don’t have a voltage rating in CIM, but you can use insulationThickness as a proxy for voltage rating in queries. Here, 5.588 mm corresponds to 220 mils, which is a common size for distribution.

Figure 28 illustrates the loads, which are called EnergyConsumer in CIM. The houses and appliances from GridLAB-D are not supported in CIM. Only ZIP loads can be represented. Further, any load schedules would have to be defined outside of CIM. Assume that the CIM loads are peak values.

Figure 29 illustrates the voltage regulator function. Note that GridLAB-D combines the regulator and transformer functions, while CIM separates them. Also, the CIM provides voltage and current transducer ratios for tap changer controls, but not for capacitor controls.

Figure 30 through Figure 32 illustrate how measurements required for RC1 can be attached to buses or other components. Individual phase measurements for voltage and capacitor status have to be added.



Figure 28: The three-phase load (aka EnergyConsumer) on bus671 is balanced and connected in delta. It has no ratedU attribute, so use the referenced BaseVoltage (Figure 13) if a voltage level is required. On the right, a three-phase wye-connected unbalanced load on bus675 is indicated by the presence of three EnergyConsumerPhase instances referencing UnbalancedLoad. For consistency in searches and visualization, UnbalancedLoad.pfixed should be the sum of the three phase values, and likewise for UnbalancedLoad.qfixed. In power flow solutions, the individual phase values would be used. Both loads share the same LoadResponse instance, which defines a constant power characteristic for both P and Q, because the percentages for constant impedance and constant current are all zero. The two other most commonly used LoadResponseCharacteristics have 100% constant current, and 100% constant impedance. Any combination can be used, and the units don’t have to be percent (i.e. use a summation to determine the denominator for normalization).



Figure 29: In CIM, the voltage regulator function is separated from the tap-changing transformer. The IEEE 13-bus system has a bank of three independent single-phase regulators at busRG60, and this example shows a RatioTapChanger attached to the regulator on phase A, represented by the TransformerTankEnd having phases=A or phases=AN. See Figure 19 for a more complete picture of TransformerTankEnds, or Figure 16 for a more complete picture of PowerTransformerEnds. Either one can be the TransformerEnd in this figure, but with a PowerTransformerEnd, all three phase taps would change in unison (i.e. they are “ganged”). Most regulator attributes of interest are found in RatioTapChanger or TapChangerControl instances. However, we need the Asset mechanism to specify ctRatio, ptRatio and ctRating values. These are inherent to the equipment, whereas the attributes of RatioTapChanger and TapChangerControl are all settings per instance. For the IEEE 13-bus example, there would be separate RatioTapChanger and TapChangerControl instances for phases B and C.



Figure 30: In CIM, the voltage measurement attaches to TopologicalNode, which we can find from the ConnectivityNode in GridAPPS-D. Positive sequence or phase A measurement is implied, so we must add a phase attribute on SvVoltage for GridAPPS-D. Physically, a voltage sensor is more closely associated with a Terminal or ConnectivityNode.



Figure 31: SvTapStep links to a TransformerEnd indirectly, through the RatioTapChanger. There is no phasing ambiguity because TransformerTankEnd has its phases attribute, while PowerTransformerEnd always includes ABC. Units for SvTapStep.position are per-unit.



Figure 32: The on/off measurement for a capacitor bank attaches directly to LinearShuntCompensator, but there is no phasing support. That needs to be proposed as a CIM extension.

# Metering Relationship to Loads in the CIM

These UML class relationships in Figure 33 through Figure 35 have not been planned for implementation in RC1, but in a future version of GridAPPS-D, they can be used to link automated meter readings with loads in the distribution system model.



Figure 33: Energy Consumers are associated to Metering Usage Points



Figure 34: Metering Usage Points have one or more EndDevices (i.e. Meters)



Figure 35: EndDevices associate to meter readings, functions and channels.

# CIM Enhancements for RC2

Possible CIM enhancements to support volt-var feeder modeling:

1. Different on and off delay parameters for RegulatingControl (Figure 5)
2. Phase modeling for EnergySource (Figure 3)
3. Current ratings for PerLengthImpedance (Figure 2). At present, some users rely on associated WireInfo, ignoring all attributes except currentRating.
4. Transducers for RegulatingControl (Figure 5)
5. Dielectric constant and soil resistivity (Figure 10)
6. Current flow and switch open/closed measurements (Figure 11)
7. Individual phase measurements for voltage and capacitor state (Figure 11)
8. Clock angles for TransformerTankEnd (i.e. move phaseAngleClock from PowerTransformerEnd to TransformerEnd (Figure 6)
9. Clarify side1 and side2 for switch phase modeling (Figure 4)

# CIM Profile in CIMTool

CIMTool was used to develop and test the profile for RC1, because it:

1. Generates SQL for the MySQL database definition
2. Validates instance files against the profile

The CIMTool developer will not be able to support the tool in future, so eventually we will use the new Schema Composer feature in Enterprise Architect.

In order to view the profile, import the archived Eclipse project *OSPRREYS\_CIMTOOL.zip* into CIMTool. Please see the CIM tutorial slides provided by Margaret Goodrich for user instructions.

Four instance files were validated against the profile in CIMTool. In order to generate them, we use a current version of OpenDSS with the *Export CDPSMcombined* command on four IEEE test feeders that come with OpenDSS:

1. **~/src/opendss/Test/IEEE13\_CDPSM.dss** is the IEEE 13-bus test feeder with per-length phase impedance matrices and a delta tertiary added to the substation transformer.
2. **~/src/opendss/Test/IEEE13\_Assets.dss** is the IEEE 13-bus test feeder with catalog data for overhead lines, cables and transformers. Capacitor controls have also been added.
3. **~/src/opendss/Distrib/IEEETestCases/8500-Node/Master.dss** is the IEEE 8500-node test feeder with balanced secondary loads.
4. **~/src/opendss/Distrib/IEEETestCases/8500-Node/Master-unbal.dss** is the IEEE 8500-node test feeder with unbalanced secondary loads.

Either the 3rd or 4th feeder will be used for the volt-var application. The 1st and 2nd feeders are used to validate more parts of the CIM profile used in RC1. In all four cases, CIMTool reports only two kinds of validation error:

1. **Isolated connectivity node**: CIMTool expects two or more Terminals per ConnectivityNode, but dead ended feeder segments will have only one on the last node. This is not really an error, at least for distribution systems.
2. **Minimum cardinality**: For TapChangerControl instances, the inherited RegulatingControl.RegulatingCondEq association is not specified. This is not really an error, as the association is only needed for shunt capacitor controls. Figure 36 shows that RegulatingCondEq was not selected for TapChangerControl in the profile, so this may reflect a defect in the validation code. Efforts to circumvent it were not successful.

With these caveats, the profile and instances validate against each other, for feeder models that solve in OpenDSS.

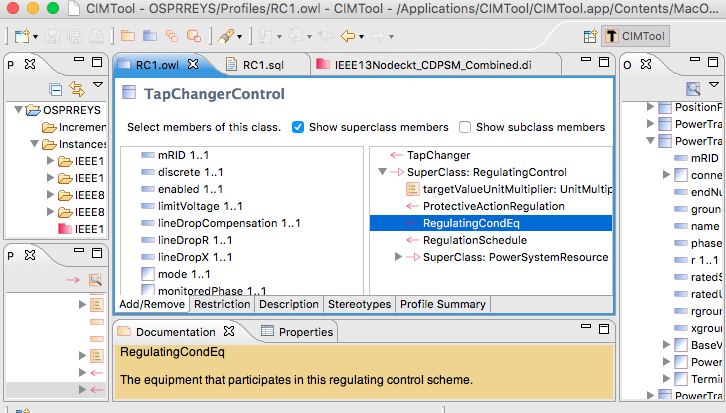


Figure 36: Profiling TapChangerControl in CIMTool; the inherited RegulatingCondEq is not included.

# Creating Data Definition Language (DDL) for MySQL

As shown at the top of Figure 36, CIMTool builds *RC1.sql* to create tables in a relational database, but the syntax doesn’t match that required for MySQL. The following manual edits were made:

1. Globally change **CHAR VARYING(30)** to  **varchar(50)** with a blank space pre-pended before the varchar
2. Globally change **“** to **`**
3. In foreign keys to enumerations, change the referenced attribute from **mRID** to **name**
4. In foreign keys to **EquipmentContainer** or **ConnectivityNodeContainer**, change the referenced table to **Line**
5. In foreign keys to **ShuntCompensator**, change the referenced table to **LinearShuntCompensator**
6. In foreign keys to **TapChanger**, change the referenced table to **RatioTapChanger**.
7. The CIM UML incorporates several polymorphic associations, which can’t be implemented directly in SQL. Base parent class tables were added for:
   1. **AssetInfo**, which can be referenced via the Parent attribute from ConcentricNeutralCableInfo, TapeShieldCableInfo, OverheadWireInfo, WireSpacingInfo, TapChangerInfo and TransformerTankInfo
   2. **TransformerEnd**, which can be referenced via the Parent attribute from PowerTransformerEnd and TransformerTankEnd
   3. **PerLengthImpedance**, which can be referenced via the Parent attribute from PerLengthSequenceImpedance and PerLengthPhaseImpedance
   4. **Switch**, which can be referenced via the SwtParent attribute from Breaker, Fuse, Sectionaliser, Recloser, Disconnector, Jumper and LoadBreakSwitch.
   5. **ConductingEquipment**, which can be referenced via the Parent attribute from ACLineSegment, EnergySource, EnergyConsumer, LinearShuntCompensator, PowerTransformer, and all of the Switch types.
8. The catalog data mechanism in Figure 8 required two new tables, one for polymorphic associations and another for many-to-many joins:
   1. **PowerSystemResource**, which can be referenced via the PSR attribute from ACLineSegment, ACLineSegmentPhase, RatioTapChanger and TransformerTank.
   2. **AssetInfoJoin**, which references AssetInfo and PowerSystemResource. This table actually supplants the Asset class in Figure 8.
9. The ShortCircuitTest in Figure 9 has a one-to-many association to TransformerEndEnfo, and we need to implement the many side by adding:
   1. **GroundedEndJoin**, which references TransformerEndInfo and ShortCircuitTest.
10. The ToTransformerEnd association in Figure 6 is one-to-many, so CIMTool did not export it to SQL. Rather than create a join table, a ToTransformerEnd attribute was added to TransformerMeshImpedance. This supports only one-to-one association, which is justified because the one-to-many case is very rare, and GridLAB-D cannot model transformers having the one-to-many association. This restriction may be removed in future versions having a semantic or graph database.

Except for the first two items, all of these adjustments arose from the absence of inheritance or polymorphism in SQL. These adjustments will make the updates, queries and views more complicated. However, they allow referential integrity to be enforced, which is one of the most important reasons to use SQL and relational databases. Other types of data store could be a more natural fit to the CIM UML, but they may not have the performance of a relational database.

In GitHub:

1. *RC1.sql* is the manually adjusted SQL export from CIMTool
2. *LoadRC1.sql* will **re-create the GridAPPS-D database in MySQL**, incorporate *RC1.sql*, and finally document the foreign keys. It should run without error.

1. See http://cimug.ucaiug.org/default.aspx and the EPRI CIM Primer at: http://www.epri.com/abstracts/Pages/ProductAbstract.aspx?ProductId=000000003002006001 [↑](#footnote-ref-1)
2. Suggest “Corporate Edition” from http://www.sparxsystems.com/ for working with CIM UML. The free CIMTool is still available at http://wiki.cimtool.org/index.h tml, but support is being phased out. [↑](#footnote-ref-2)
3. OSPRREYS is an older name for GridAPPS-D [↑](#footnote-ref-3)
4. https://github.com/GRIDAPPSD/Powergrid-Models/CIM [↑](#footnote-ref-4)