# CIM Profile and Database for OSPRREYS RC1

This document summarizes the use of a reduced-order CIM to support feeder modeling for the volt-var application in 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. The three main sections of this document cover:

1. Diagrams and explanation of the reduced-order CIM
2. Generating and testing the CIM profile in CIMTool
3. Generating and tuning the DDL file that creates a MySQL database

## Reduced-Order CIM for RC1

Figures 1-11 present the UML diagrams generated from Enterprise Architect, build 1308. 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:

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* from GitHub for the latest updates.

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 next 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.

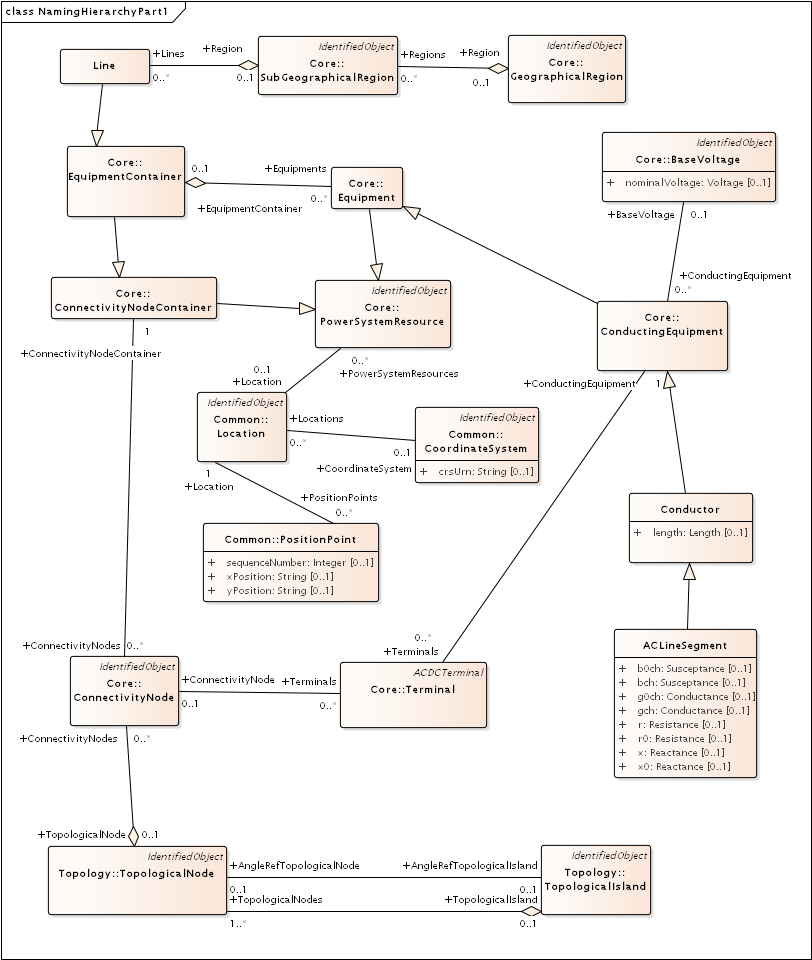


Figure 1: Placement of ACLineSegment into a Line (aka Feeder). In OSPRREYS, the Line is the EquipmentContainer for all power system components and the ConnectivityNodeContainer for all nodes. It also corresponds to one TopologicalIsland. It’s part of a SubGeographicalRegion and GeographicalRegion for 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.

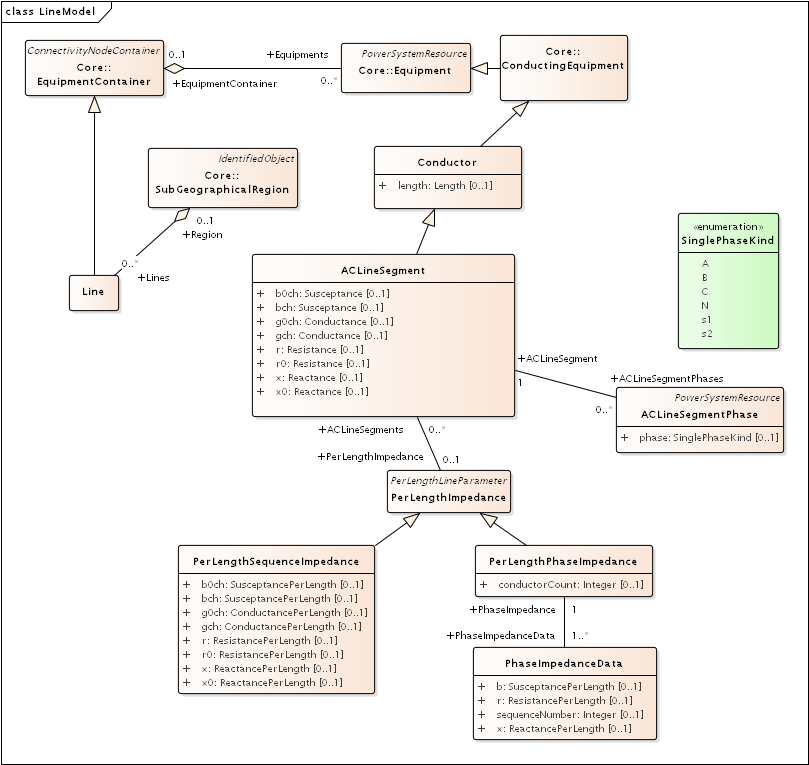


Figure 2: There are four different ways to specify ACLineSegment impedances. In all cases, Conductor:length is required. The first way is to specify the individual ACLineSegment attributes, which are sequence impedances and admittances, leaving PerLengthImpedance null. The second way is to specify the 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.

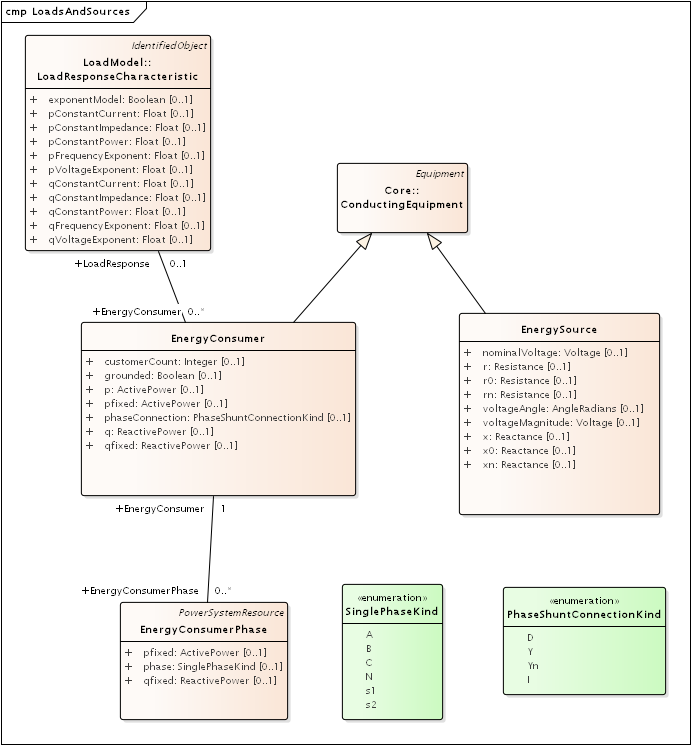


Figure 3: The EnergySource is balanced three-phase, representing a transmission system source (this is probably not the way we’ll model distributed generation in future versions). The EnergyConsumer is a ZIP load, possibly unbalanced, with an associated LoadResponse instance defining the ZIP coefficients. 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).

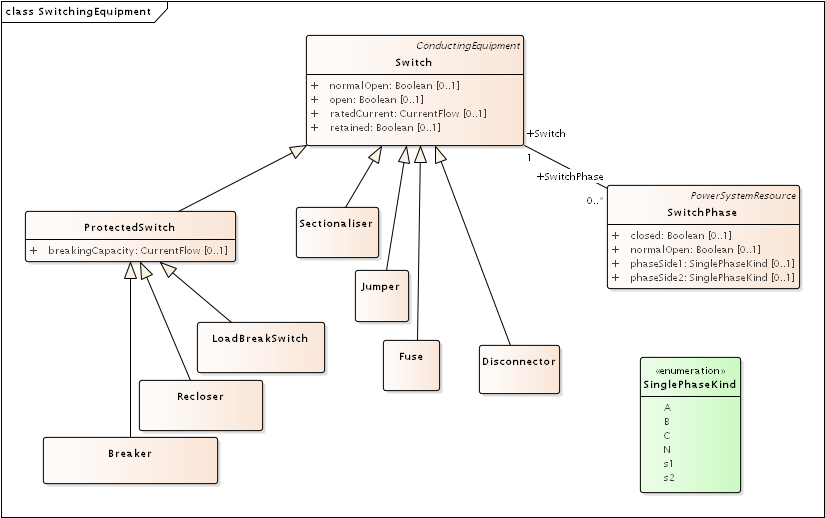


Figure 4: There are seven different kinds of Switch supported in the CIM, and all of them have zero impedance. They would all behave the same in power flow analysis, and all would require many more attributes than are defined in CIM to support protection analysis. The use cases for SwitchPhase include 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.

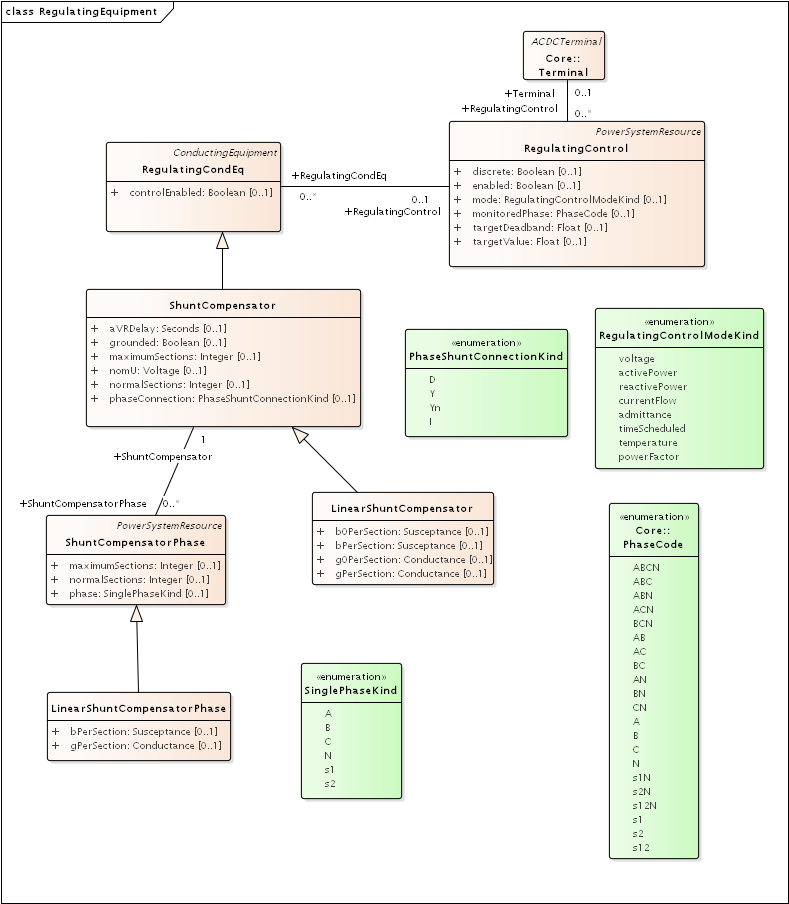


Figure 5: On the left, LinearShuntCompensator and LinearShuntCompensatorPhase define capacitor banks, in a way very similar to EnergyConsumer in Figure 3. The kVAR ratings must be converted to susceptance based on the nominal voltage, nomU. Note that aVRDelay is really a capacitor control parameter, 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.

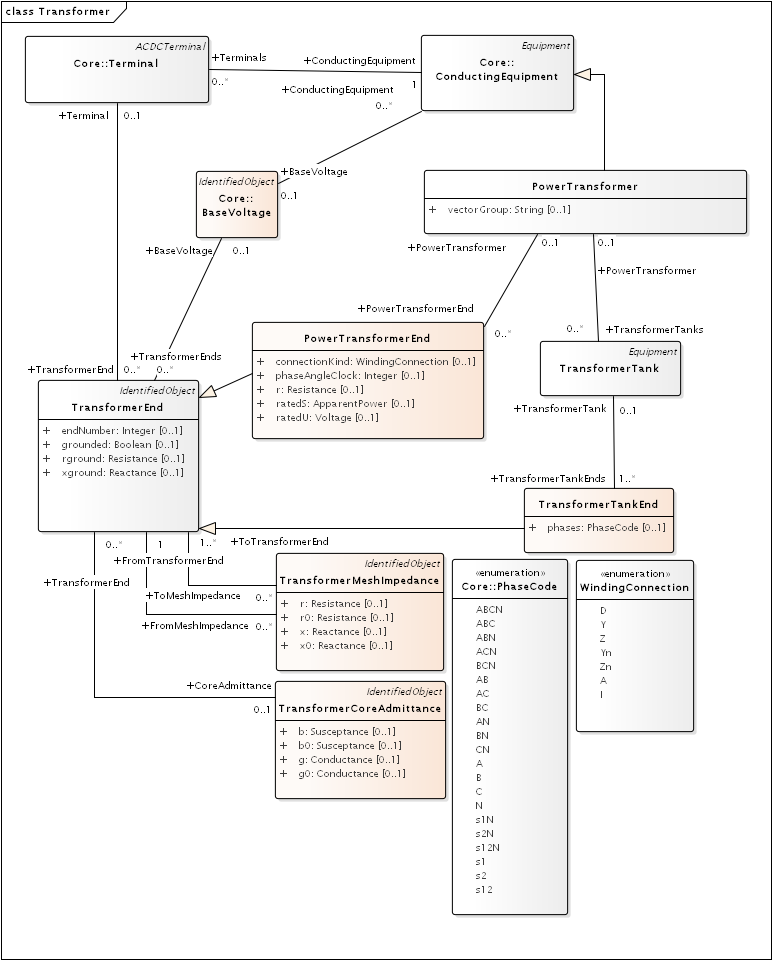


Figure 6: PowerTransformers may be modeled with or without tanks, and in both cases vectorGroup should be specified according to IEC transformer standards (e.g. Dy1 for many substation transformers). The case without tanks is most suitable for balanced three-phase transformers that won’t reference catalog 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.

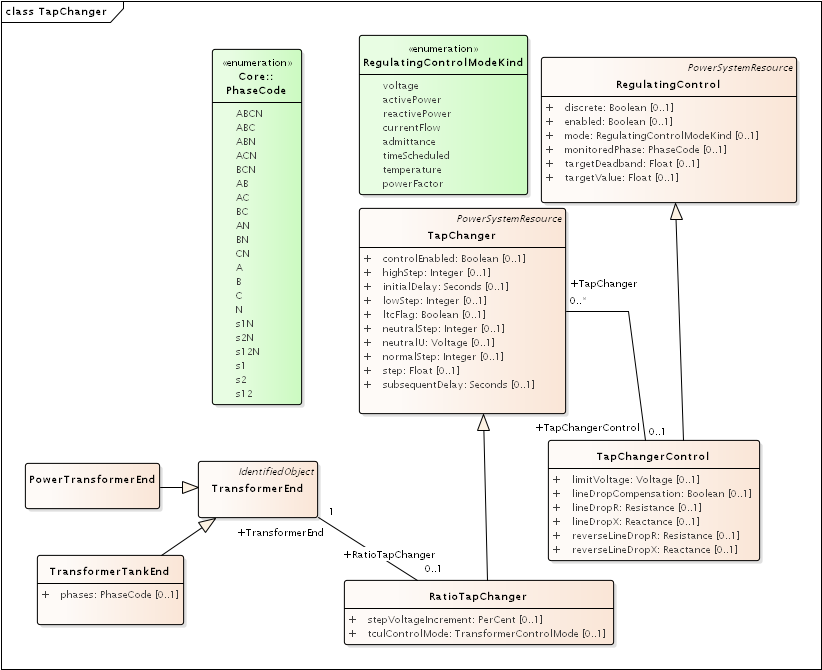


Figure 7: A RatioTapChanger can represent a transformer tap changer on the associated TransformerEnd. The RatioTapChanger has some parameters defined in a direct-associated TapChangerControl, which inherits from RegulatingControl some of the same attributes used in capacitor controls (Figure 5). Therefore, 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.

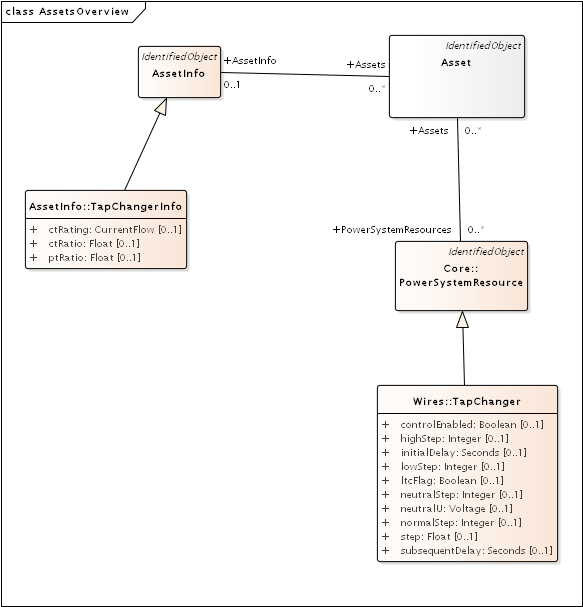


Figure 8: Many distribution software packages use the concept of catalog data, aka library data, especially for lines and transformers. We use the Asset and AssetInfo packages to implement this in CIM. Here, the TapChangerInfo class includes the CT rating, CT ratio and PT ratio parameters needed for 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.

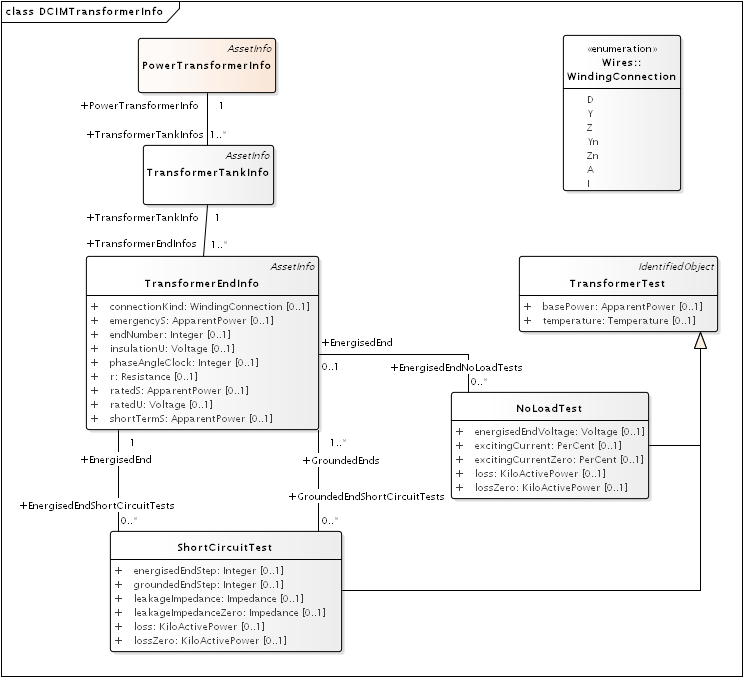


Figure 9: The catalog mechanism for transformers will associate a TransformerTank (Figure 6) with TransformerTankInfo (here), via the one-to-many mechanism described in Figure 8. The PowerTransformerInfo collects TransformerTankInfo by reverse association, but it does not link with PowerTransformer. 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.

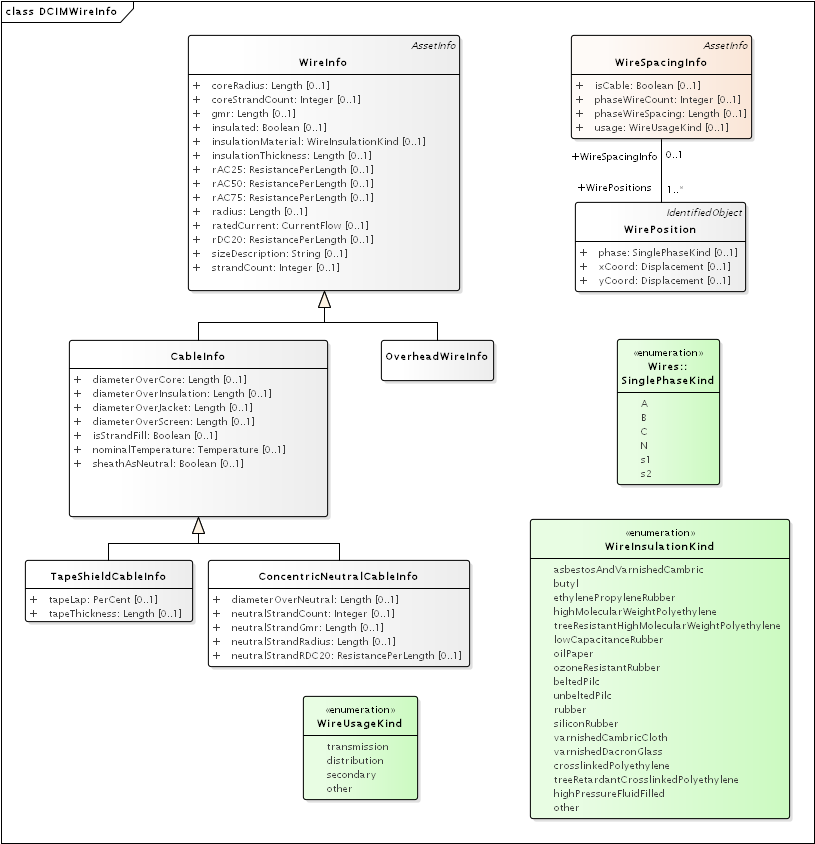


Figure 10: The catalog / library mechanism for ACLineSegment will have a WireSpacingInfo associated as in Figure 9. This will indicate whether the line is overhead or underground. phaseWireCount and phaseWireSpacing define optional bundling, so these will be 1 and 0 for distribution. The number of phase 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.

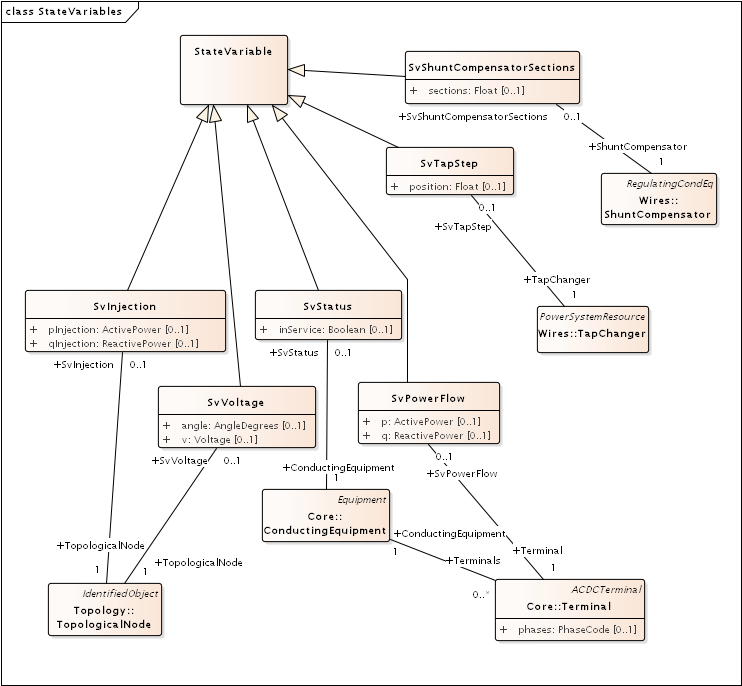


Figure 11: The CIM state variables package might be used to mimic sensor locations and values on the distribution system. Voltages are measured on TopologicalNodes, power flows are measured at Terminals, step positions are measured on TapChangers, status is measured on ConductingEquipment, and on/off 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 OSPRREYS.

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)
4. Transducers for RegulatingControl (Figure 5)
5. Dielectric constant and soil resistivity (Figure 10)
6. Current and switch open/closed measurements (Figure 11)

## 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 12 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.

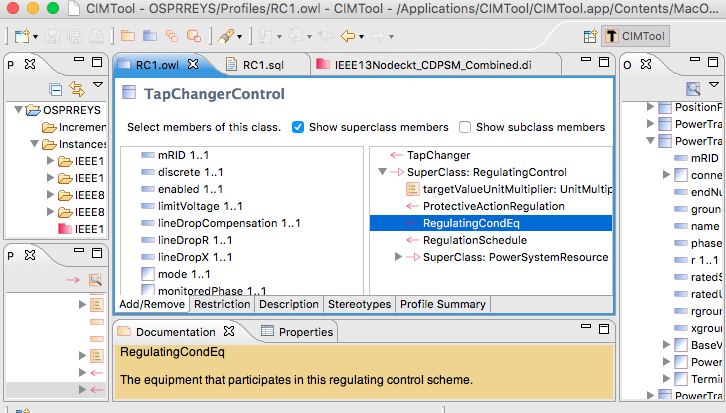


Figure 12: Profiling TapChangerControl in CIMTool; the inherited RegulatingCondEq is not included.

## Creating Data Definition Language (DDL) for MySQL

As shown at the top of Figure 12, 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 OSPRREYS database in MySQL**, incorporate *RC1.sql*, and finally document the foreign keys. It should run without error.