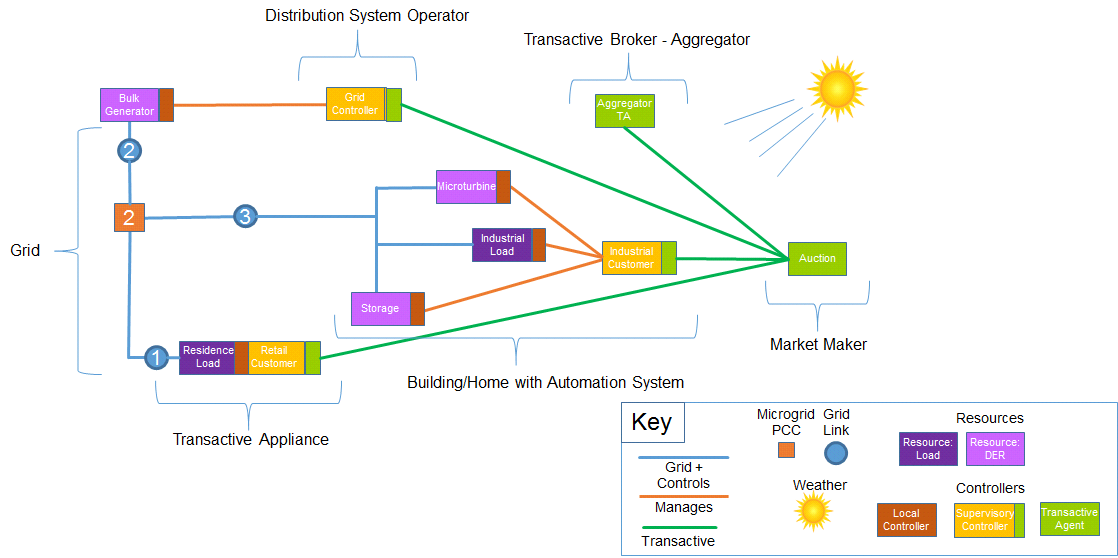
The Transactive Energy Challenge Abstract Component Model

# Overview

This model was the product of a "Tiger Team Effort" engaged by NIST, PNNL, Vanderbilt, and CMU/MIT during the summer of 2016 in support of the NIST Transactive Energy Challenge <https://pages.nist.gov/TEChallenge/>. The purpose of this activity was to distill from the collective experience of the participants an abstract model of a Transactive Energy system consisting of an energy grid, loads, generators, controllers, and transactive agents.

The tiger team created an Abstract Component Model, which provides a basis for common discussions, similar to the IEEE standard feeder models that the industry has used for more than decade to test changes to simulations. The feeder models have allowed apples to apples comparisons to happen, speeding up the development of better simulation tools, and understanding of modeling and simulation in most power engineering programs. The Extensible Component Model should be able to provide similar support for TE in the industry. A notional diagram of the model is below.



1. Notional Topology

While the diagram shows a system that is much simpler than a real-world situation, it is complete from a testing point of view. There is at least one actor of each major type and one component of each major type. The model can be used to evaluate any of the SGIP TECG use cases (<http://www.sgip.org/wp-content/uploads/SGIP_White_Paper_TE_Application_Landscape_Scenarios_12-15-2016_FINAL.pdf>), and any other use cases that support TE. The diagram illustrates the variations in realizations of components of a transactive energy system. The model in this document goes on to produce a set of model components that can be assembled and otherwise extended to simulate transactive systems of arbitrary design.

The goal of the model is to be able to understand, discuss, evaluate and validate any TE approach. Additionally, the model can be used to look at grid operations and controls as part of the TE approach. In a teaching environment, it can be used to test the student’s understanding and to frame the project that the student is working on within the TE context.

To understand the model, consider that there are building blocks of resources, controllers, transactive agents, the grid itself. These can be composed into various devices and systems. For example, a transactive appliance contains a load, a local controller, a supervisory controller, and a transactive agent. This composite device can participate in a transactive energy negotiation and regulate its operation accordingly to price signals.

Obviously, TE is a complex system of systems problem and the detailed simulations will require a diverse set of simulation tools. The model allows a connection to be determined between tools that might be simulating a building management system, or a storage control system and the rest of the TE environment. Allowing specific tools, models, algorithms, and behavior on the balance of the TE environment. The model allows the creating of test of end to end security for each transaction both for control and communications.

Control simulations and algorithms can be layered on the TE model for testing. Control transactions that may be needed to balance the network can be layered on to the TE model to determine reactions and needed latency, or lead times. Pricing and economics can be discussed and tested within the TE model, to determine supporting transactions. With the right economic simulations installed in the model, economic simulation can be tested. With enough customer behavior information, the simulations can simulate within the TE model how customers would react and the trigger prices that might be needed to achieve desired behavior. In short, the TE model provides a simplified environment to talk about, design and test almost any aspect of a TE approach. Note that this abstract component model does not specify an implementation. However, it provides the skeleton within which any given implementation can be discussed and compared on similar terms.

To do this, the tiger team created a common platform with well-defined interfaces and semantics that stakeholders can understand and use to evaluate their own situations in their own context. The TE model makes it possible for stakeholders that do not understand the underlying grid model, or other specific technical aspects to quickly understand TE and evaluate those items that are important to them without having to have an expert help them setup and evaluate their specific aspect of TE.

The overall TE model allows stakeholders to not only test items, but to design specific algorithms and tools that are proprietary. That means that competitors can use the same model to each test their specific competitive advantage and keep it secret from the competition, but at the same time be able to discuss in general terms what they are attempting to achieve with other stakeholders in a context that everyone understands.

In the long run, data sets, behavior models, common starting tools, message sets, communications modeling tools and other supporting tools will be developed on the TE model so that any stakeholder has a starting toolbox from which to tinker. Then, stakeholders can create their own improved tools for the specific areas they are interested in, while running the common tools for other aspects. This shortens development times for stakeholders and greatly lowers the barriers to entry for involvement.

Obviously, the TE model can be extended by any stakeholder in any specific fashion to deal with simulation of that aspect. For instance the TE model might be extended to all the appliances and other devices in a home for a stakeholder looking at home energy management systems.

The implementation has to faithfully implement a minimum set of specific interfaces. These interfaces can be extended as needed by the implementer. At the same time complexity can be hidden by combining components in the model where the interfaces between the components are not important to the question(s) being evaluated.

The implementation needs to be able to orchestrate a set of components. These components, like the interfaces can be extended or minimized depending on the question(s) in play. Ideally the components can be simulated by the same experiment controller.

The implementation also needs to support the defined set of grid nodes, resources, controllers, transactive agents, and market simulations to provide the required comparison baseline.

There is also a need to implement a core set of analytics that will be able to evaluate the data from the implemented model against the baseline model and data.

The detailed technical specification herein has been designed to be implemented on one or more simulation platforms. Many of the commonly used simulation platforms will support the implementation.

The balance of this model document is organized into the following sections:

Section 2: Model -- the description of the core component model itself

Section 3: Scenario -- the orchestration of the model into a "canonical experiment" which enables components implemented and designed to support the abstract interfaces to work in concert in a simulation.

Section 4: Beta Use Case -- a reference use case designed by researchers at PNNL that represent an example instance of the component model

Section 5: Composite and Extended Classes -- some examples of how to merge and/or extend the classes to produce instance models shown in figure 1 above.

# Model

The Model consists of a set of abstract components for use in studying transactive energy. Each model represents a set of roles or interfaces that an actual device or computing platform might play in a transactive energy simulation. This section is divided into three parts:

* The core transactive component models
* The interfaces that can be realized for interacting with the components
* The data types that flesh out the minimum detailed attributes that can be exchanged by the components

## Transactive Energy Components

These model components expose the key interfaces of the TE Common Model platform. Shown are the roles, their interfaces, and persistent data required (Note that in an actual implementation, several of these "roles" may be combined into a single instance exposing multiple interfaces. See examples in the "Composite Classes" section).

This section will describe the core components of the model.

At the heart of the model is a simulation grid which represents the energy distribution system. There is a great deal of modeling and experience in describing grids. However, for the purpose of this transactive energy abstraction, the grid represents the entirety of connected devices responsible for delivery and operation. It only exposes the links and nodes to which transactive energy resources are attached. Resources consist of loads and generators (and also storage) devices that can sink or source energy.

There are two types of controllers -- local and supervisory. The local controller is a component that understands the nature of a resource. For example a thermostat is a good example of a local controller for an HVAC system load. The component model concentrates the physical nature of resources in the resource and the logical part of the component in the local controller. This allows the "physical part" to interact with the physics of the grid simulation while the local controller can interact with the supervisory controller in a higher abstraction of supervisory control.

There is a transactive agent that is typically tightly coupled to the supervisory controllers which is responsible for offering, bidding, and negotiating the price of energy.

Finally, a weather component is responsible for providing the changing environment that drives energy production and consumption.

Two additional meta components -- the experiment manager, and, the analytics represent the simulation test harness that orchestrates and analyzes the transactive energy scheme.

### Transactive Energy Components diagram

This diagram illustrates the classes or roles of components of the TE Challenge Abstract Component Model. The diagram shows three groupings of model components:

Core Components -- these represent the granularity of components that can be used in simulations. Additionally, they can be combined to represent less-granular components by aggregating their function and interfaces into composite components.

Specializations -- these represent specializations of the core components for specific roles in transactive energy simulations.

Experiment Orchestration and Analysis -- these represent the orchestrator for the simulation experiment (Experiment Manager), and, the core Analytics component that evaluate the data produced during the simulation.

Note that these models provide for a minimum of interoperability and the ability to define an experiment that can exercise the models based on these designs. Any particular realization of these models might extend their capabilities and information exchanged. The base interoperable characteristics herein provide for the consistency of simulation execution and minimal availability of data for the analytics.



1. Transactive Energy Components

### Auction

A specialization of the TransactiveAgent that is essentially the market maker or broker. Some transactive energy schemes are purely peer to peer. They do not need an auction component. Others require a central component where participants can contribute their offerings and bids and that conducts the algorithm by which pricing is determined from the collective participants.

### Weather

The Weather component provides environmental conditions during a TE experiment. The weather component itself is based on the National Solar Radiation Data Base from NREL know as Typical Model Year (TMY)3 <http://rredc.nrel.gov/solar/old_data/nsrdb/1991-2005/tmy3/>

| **Name** | **Type** | **Notes** |
| --- | --- | --- |
| date | char | mm/dd/yyyy |
| time | char | hh:mm:ss local time |
| extra\_terrestrial\_radiation\_normal | double | (W/m^2) |
| extra\_terrestrial\_radiation | double | (W/m^2) |
| global\_horizontal\_irradiance | double | (W/m^2) |
| global\_horizontal\_irradiance\_source | char | GHI source |
| global\_horizontal\_irradiance\_uncertainty | double | GHI uncert (%) |
| direct\_normal\_irradiance | double | DNI (W/m^2) |
| diffuse\_horizontal\_irradiance\_source | char | DHI source |
| diffuse\_horizontal\_irradiance\_uncertainty | double | DHI uncert (%) |
| global\_horizontal\_illuminance | double | GH illum (lx) |
| global\_horizontal\_illuminance\_source | char | GH illum source |
| global\_horizontal\_illuminance\_uncertainty | double | Global illum uncert (%) |
| direct\_normal\_illuminance | double | DN illum (lx) |
| direct\_normal\_illuminance\_source | char | DN illum source |
| direct\_normal\_illuminance\_uncertainty | double | DN illum uncert (%) |
| diffuse\_horizontal\_illuminance | double | DH illum (lx) |
| diffuse\_horizontal\_illuminance\_source | char | DH illum source |
| diffuse\_horizontal\_illuminance\_uncertainty | double | DH illum uncert (%) |
| zenith\_luminance | double | Zenith lum (cd/m^2) |
| zenith\_luminance\_source | char | Zenith lum source |
| zenith\_luminance\_uncertainty | double | Zenith lum uncert (%) |
| total\_sky\_cover | double | TotCld (tenths) |
| total\_sky\_cover\_source | char | TotCld source |
| total\_sky\_cover\_uncertainty | double | TotCld uncert (code) |
| opaque\_sky\_cover | double | OpqCld (tenths) |
| opaque\_sky\_cover\_source | char | OpqCld source |
| opaque\_sky\_cover\_uncertainty | double | OpqCld uncert (code) |
| dry\_bulb\_temperature | double | Dry-bulb (C) |
| dry\_bulb\_temperature\_source | char | Dry-bulb source |
| dry\_bulb\_temperature\_uncertainty | double | Dry-bulb uncert (code) |
| dew\_point\_temperature | double | Dew-point (C) |
| dew\_point\_temperature\_source | char | Dew-point source |
| dew\_point\_temperature\_uncertainty | double | Dew-point uncert (code) |
| relative\_humidity | double | RHum (%) |
| relative\_humidity\_source | char | RHum source |
| relative\_humidity\_uncertainty | double | RHum uncert (code) |
| pressure | double | Pressure (mbar) |
| pressure\_source | char | Pressure source |
| pressure\_uncertainty | double | Pressure uncert (code) |
| wind\_direction | double | Wdir (degrees) |
| wind\_direction\_source | char | Wdir source |
| wind\_direction\_ncertainty | double | Wdir uncert (code) |
| wind\_speed | double | Wspd (m/s) |
| wind\_speed\_source | char | Wspd source |
| wind\_speed\_uncertainty | double | Wspd uncert (code) |
| horizontal\_visibility | double | Hvis (m) |
| horizontal\_visibility\_source | char | Hvis source |
| horizontal\_visibility\_uncertainty | double | Hvis uncert (code) |
| ceiling\_height | double | CeilHgt (m) |
| ceiling\_height\_source | char | CeilHgt source |
| ceiling\_height\_uncertainty | double | CeilHgt uncert (code) |
| precipitable\_water | double | Pwat (cm) |
| precipitable\_water\_source | char | Pwat source |
| precipitable\_water\_uncertainty | double | Pwat uncert (code) |
| aerosol\_optical\_depth | double | AOD (unitless) |
| aerosol\_optical\_depth\_source | char | AOD source |
| aerosol\_optical\_depth\_uncertainty | double | AOD uncert (code) |
| albedo | double | Alb (unitless) |
| albedo\_source | char | Alb source |
| albedo\_uncertainty | double | Alb uncert (code) |
| liquid\_precipitation\_depth | double | Lprecip depth (mm) |
| liquid\_precipitation\_quantity | double | Lprecip quantity (hr) |
| liquid\_precipitation\_depth\_source | char | Lprecip source |
| liquid\_precipitation\_depth\_uncertainty | double | Lprecip uncert (code) |
| direct\_normal\_irradiance\_source | char | DNI source |
| present\_weather | double | PresWth (METAR code) |
| direct\_normal\_irradiance\_uncertainty | double | DNI uncert (%) |
| present\_weather\_source | char | PresWth source |
| diffuse\_horizontal\_irradiance | double | DHI (W/m^2) |
| present\_weather\_uncertainty | double | PresWth uncert (code) |
| station\_id\_code | int | station identifier code |
| station\_name | char |  |
| station\_state | char |  |
| time\_zone | double |  |
| atitude | double |  |
| longitude | double | station longitude |
| elevation | double |  |

### Grid

A simulation of a power grid or grid segment.

A Grid consists of a set of Link structures that represent a network of interconnected nodes that comprise and energy system. Resources such as loads and generators are "attached" to the nodes of the Grid.

| **Name** | **Type** | **Notes** |
| --- | --- | --- |
| Nodes | Link | List of nodes in Grid |

### LocalController

A simple base controller that knows how to control a resource based on a dimensionless modulation setting received from a SupervisoryController. It does not have awareness of any other component of the system.

These include logic for such things as voltage regulators or protection devices and determine what state the device should be in and how it might progress to the next state. Additionally, thermostats, loop controllers, etc....

| **Name** | **Type** | **Notes** |
| --- | --- | --- |
| actualDemand | float | This attribute defines the power being consumed by the device (as measured by the present subinterval demand) at the present time. |
| demandLimits | PowerRatings | The operational demand characteristics and their associated end points for the load. |
| downRamp | PowerRampSegmentType | This attribute defines the reduction in power over time when a load being partially or fully de-energized has a complex load reduction profile. For each element of the Load.downRamp array, the downRamp[n].rate defines the amount of power decrease and the downRamp[n].duration defines the length of time in seconds upon which the decrease is in effect. If the downRamp[n].beginRamp attribute is defined for a ramp segment, this is the initial value of the ramp segment; if it is not present the initial value of the ramp equals the ending value of the previous ramp segment.  Although the downRamp attribute name implies that the rise is monotonically decreasing, individual array elements may have slopes less than, greater than, or equal to 0. However, the overall trend of the function shall be decreasing.  The downRamp function shall measure the time from the load being fully energized until the power is completely depleted. If downRamp is not present, the power decrease to 0 shall be instantaneous.  When a curtailable load is partially curtailed (less than the maximumCurtailableDemand) and curtailment is increased, the power will decrease starting at n-th downRamp of the sum from 0 to n of the downRamp[n].rate that is closest to the actualCurtailedDemand. |
| upRamp | PowerRampSegmentType | This attribute defines the increase in power over time when a load being partially or fully energized has a complex demand restoration profile. For each element of the Load.upRamp array, the upRamp[n].rate defines the amount of power increase and the upRamp[n].duration defines the length of time in seconds upon which the increase is in effect. If the upRamp[n].beginRamp attribute is defined for a ramp segment, this is the initial value of the ramp segment; if it is not present the initial value of the ramp equals the ending value of the previous ramp segment.  Although the upRamp attribute name implies that the rise is monotonically increasing, individual array elements may have slopes less than, greater than, or equal to 0. However, the overall trend of the function shall be increasing.  The upRamp function shall measure the time from the load being fully de-energized until the power is completely restored as defined by Load.maximumDemand. If this attribute is not present, the power increase upon restoration shall be instantaneous.  When a curtailable load is partially curtailed (less than the maximumCurtailableDemand) and curtailment is reduced, the power will increase starting at n-th recoveryRamp of the sum from 0 to n of the upRamp[n].rate that is closest to the actualCurtailedDemand. |
| locked | Boolean | This attribute defines whether the load is locked and therefore ineligible for curtailment; or unlocked and available for curtailment.  Load locking behavior changes depending on the load's curtailmentStatus attribute value at the time the load was locked. If the Load.locked attribute is set to TRUE and the:  1) load is not curtailable, the load is immediately locked. The behavior of this operation is a local matter;  2) curtailmentStatus is curtailmentInactive, the load will immediately be locked out from curtailment eligibility;  3) curtailmentStatus is curtailmentNoncompliant, the load will cycle to its curtailmentInactive state then immediately be locked out of curtailment eligibility;  4) If the CurtailableLoad supports multi-stage curtailment, the load will cycle to its curtailmentInactive state for the present curtailment stage and then be locked out of any further curtailment eligibility.  5) curtailmentStatus is curtailmentNoncompliant, the load will cycle to its curtailmentInactive state then immediately be locked out of curtailment eligibility;  6) If the CurtailableLoad supports multi-stage curtailment, the load will cycle to its curtailmentInactive state for the present curtailment stage and then be locked out of any further curtailment eligibility.  Loads that are locked will remain in the locked state indefinitely until the Load.locked attribute is reset to FALSE. The mechanism used to unlock the load is a local matter. |
| status | LoadStatusType | This attribute defines the current status of the load. For non-curtailable loads, it provides the present communication status and reliability of the data. For curtailable loads, it also defines if the load is eligible for curtailment or why it is ineligible for curtailment. |

### Resource

Grid connected Resources can be loads, generators, storage devices. They consume or generate energy.

Resources are considered to be intelligent in that they can provide information about themselves and have a defined interface for local control over their operation.

In practice, there may be many details of a resource that a modeler may expose. Those shown in this model are minimum interfaces required to perform standardized simulations for transactive energy. Actual models will inherit from these more general interfaces to include the specialized behaviors and information exchanges.

| **Name** | **Type** | **Notes** |
| --- | --- | --- |
| gridNodeId | GridNodeId | Identifies grid node load is connected to. |
| current | Current | Power |
| power | Power | rate of change of power |
| impedance | Impedance | Energy over time step |
| phases | Phases |  |
| voltage | Voltage |  |
| status | boolean |  |

### SupervisoryController

Individual agents that control Resources; this does not need to be a one-to-one mapping of controller to resource (e.g., this may include a non-transactive aggregator or volt-var control system).

The SupervisoryController acts by providing a dimensionless command (and other data in an extended class if so implemented) to a LocalController. Note it is the LocalController that knows the details of the Resource to be controlled.

This explicitly does not contain transactive elements, but does contain non-transactive optimizations and operator and consumer actions. NOTE: this element will need to be extremely broad, since there are hundreds of different control variables that one may need access to.

For example, a building control system which has setpoints and schedules, and other customer preferences may act as the SupervisoryController for all Resources in a building or campus.

| **Name** | **Type** | **Notes** |
| --- | --- | --- |
| resources | Resource |  |

### BaseModelComponent

General Transactive Energy Model Component. This abstract component provides for the initialization of simulation model components.

| **Name** | **Type** | **Notes** |
| --- | --- | --- |
| name | char | Name of the component |
| description | char |  |

### TransactiveAgent

Describes a transaction to occur at a given place, including an expression of value, logic for estimating that value (e.g., forecasts) and quantity (including limits), and rules for how “bids” are formed and how often they are presented. This would include all “market” functions including device level bidding (replacing a traditional controller/thermostat), large-scale optimization (e.g., it could be used to describe an ISO or double-auction), etc.

### GridController

A specialization of the SupervisoryController that provides for supervisory control of the grid segment and represents the distribution system operator (DSO) or similar entity that can represent the grid management in a transactive energy scheme.

### Load

A specialization of a resource that represents a customer premise based load.

### Generator

A specialization of a resource that represents a customer premise based grid connected generation source.

### ExperimentManager

The ExperimentManager runs the experiment. It is typically responsible for providing initialization data for the individual components and managing the time progression of an experiment.

### Analytics

Analyzes data and produces metrics of the scenario. Each instance of Analytics is generated for a step in the time sequence of the experiment based on the timing of messaging for each phase of the message loop.

| **Name** | **Type** | **Notes** |
| --- | --- | --- |
| gridPower | Power | Power provided by the Grid. |
| loadProfile | Energy | Energy consumed by each load. |
| generationProfile | Energy | Generation by generator. |
| aggregatedLoadsByHousehold | Energy | Aggregated load by household. |
| priceNegotiations | Tender | Sequence of all tenders. |
| realizeMarketPricing | Quote | Realized Market price quotes. |
| voltage | Voltage | Voltage at every link point in pairs in link order and fromVoltage prior to toVoltage. |

## Interfaces

The interfaces for the model represent those messages that are received (subscribed). It is assumed that in order to invoke these interfaces, the source can publish the data.

In implementing this model, pub-sub or request-response can produce equivalent results and the arrows and data flowed interpreted appropriately to the message mechanism.

### Interfaces diagram

This diagram illustrates the interfaces defined for the TE Challenge Abstract Component Model. Shown are the core components of the model and those interfaces that they realize.

Note that each interface exchanges a named type data structure. The details of this data can be found in the Data Types section which follows.



1. Interfaces

## Data Types

Data Types used in message exchanges. This section details the attributes used in interface exchange and class definitions.

### DataTypes diagram

This diagram presents data types in three categories:

Component Model Data Structures -- defines the main data structures that represent grid state.

Interface Parameter Classes -- defines the content of interface messages.

Data Descriptions Imported from ASHRAE 201 -- FSGIM. These data definitions were modified from the standard to fit the data primitives of this modeling effort. They can be losslessly converted from FSGIM data types.



1. DataTypes

### Energy

A complex vector of energy by phase.

| **Name** | **Type** | **Notes** |
| --- | --- | --- |
| energyByPhase | ComplexNumber |  |

### AttachNodeDescription

Parameters to attach a node to the Grid.

| **Name** | **Type** | **Notes** |
| --- | --- | --- |
| gridNodeID | GridNodeId | Node identifier to attach to in the Grid model. |
| impedance | Impedance | Impedance matrix for phase connections. |
| phases | Phases | Phases for attachment. |

### ComplexNumber

A complex number

| **Name** | **Type** | **Notes** |
| --- | --- | --- |
| real | float | The real part. |
| imaginary | float | The imaginary part. |

### Current

| **Name** | **Type** | **Notes** |
| --- | --- | --- |
| currentByPhase | ComplexNumber |  |

### GridControl

GridControl service data structure.

### GridVoltageState

| **Name** | **Type** | **Notes** |
| --- | --- | --- |
| voltage | Voltage |  |
| phases | Phases |  |

### Impedance

| **Name** | **Type** | **Notes** |
| --- | --- | --- |
| impedanceByPhase | ComplexNumber |  |

### Link

Describes the physical components of the system and their internal state properties, such as power (or current) flow, current tap position, etc. It also includes topological information, such as “to” and “from”, which makes the nodal information implicit. Note that "to" and "from" simply identify two ends of the link and do not make a statement about the direction of flow of power/energy.

| **Name** | **Type** | **Notes** |
| --- | --- | --- |
| id | char | link segment id. |
| name | char |  |
| fromId | char | node 0 id |
| toId | char | node 1 id |
| length | float | length of link segment in meters |
| status | boolean | true if connectivity established  false if connectivity denied |
| impedance | ComplexNumber | matrix of impedances for each phase |
| phases | char | sequence of phases from A, B, C, N, L1, L2. e.g. three phase 4 wire -- ABCN  Note phase order indicates the index in to the vector array of impedances or power or voltage. Voltage relative to N. |
| powerTo | ComplexNumber | Power into the "to" node. |
| powerFrom | ComplexNumber | Power into the "from" node. |
| currentFrom | Current | Current flowing into the From node. |
| currentTo | Current | Current flowing into the To node. |
| voltageFrom | Voltage | Voltages at From node. |
| voltageTo | Voltage | Voltages at To node. |

### PiecewiseLinearSegment

The PiecewiseLinearSegment class defines the attributes needed to specify a single straight line segment for a piecewise linear curve. Each straight line segment is specified by two X-axis coordinates, percentOfFullRatedOutputBegin and percentOfFullRatedOutputEnd; and by two Y-axis coordinates, percentOfFullRatedInputPowerDrawnBegin and percentOfFullRatedInputPowerDrawnEnd.

The entire piecewise linear curve is defined by the runningProfile attribute; where the 'percent of full rated input power' is a function of the 'percent of full rated output'. That is, as the output varies between 0 and 100 percent; the function maps to the percentage of input power (0..100) required to achieve the specified output.

| **Name** | **Type** | **Notes** |
| --- | --- | --- |
| desiredFractionOfFullRatedOutputBegin | float | This attribute defines the starting x-coordinate of the straight line segment. |
| desiredFractionOfFullRatedOutputEnd | float | This attribute defines the ending x-coordinate of the straight line segment. |
| requiredFractionOfFullRatedInputPowerDrawnBegin | float | This attribute defines the starting y-coordinate of the straight line segment. |
| requiredFractionOfFullRatedInputPowerDrawnEnd | float | This attribute defines the ending y-coordinate of the straight line segment. |

### Power

A complex vector of power by phase.

| **Name** | **Type** | **Notes** |
| --- | --- | --- |
| powerByPhase | ComplexNumber |  |

### PowerCurve

This class describes the characteristics of a mathematical function used to estimate the power consuming characteristics of a load or the power generating characteristics of a generator.

| **Name** | **Type** | **Notes** |
| --- | --- | --- |
| realPowerCurve | PiecewiseLinearSegment | This attribute defines the real component of a single piecewise linear curve mapping the percentage of power consumed by the device as a function of the present level of operation of the device. |
| reactivelPowerCurve | PiecewiseLinearSegment | This attribute defines the reactive component of a single piecewise linear curve mapping the percentage of power consumed by the device as a function of the present level of operation of the device. |
| maximumRealPower | float | This attribute defines the maximum real power consumed (or supplied) by the device in units specified in PowerRealType. |
| maximumReactivePower | float | This attribute defines the maximum reactive power consumed (or supplied) by the device in units specified in PowerReactiveType. |

### PowerRampSegmentType

The PowerRampSegmentType data structure is used to define a single array element of the recoveryRamp and stagingRamp array of the Load class. Each array element defines the beginning demand for the line segment and the rate of rise or drop. These attributes combined with the duration completely forms a line segment defining a portion of the ramp.

| **Name** | **Type** | **Notes** |
| --- | --- | --- |
| beginRamp | float | The attribute defines the quantity of power at the start of the ramp segment. If this attribute is not defined in the segment, the start of the ramp is assumed to be the end of the ramp of the previous segment. |
| duration | int | The attribute defines the time horizon in seconds upon which the associated rise or drop is valid. |
| rampToCompletion | boolean | The attribute defines whether the ramping up or down of this load may be halted in midstream (false) or once started must complete through all segments of the ramp (true). As an example, a multistate fan may only use a portion of the ramp, as it sequences from low to medium to high speed levels (false); whereas, a production line, once started, may need to run through its complete set of ramp segments (true). If the attribute is not defined it is assumed to be false. |
| rate | float | The attribute defines rate of rise (positive value) in demand or the rate of drop (negative value) in demand when a load either powers up or shuts down respectively. Its sister attribute, duration, defines the time frame upon which the rate is defined. |

### PowerRatings

This class describes the power characteristics of a Load (or Generator) component. The attributes defined allow specifying the minimum and maximum expected power draw from the load (supply from a generator). It also allows a series of predefined operation power curves to be defined with one designated as presently being operational.

| **Name** | **Type** | **Notes** |
| --- | --- | --- |
| adjustedNoDRPower | float | This attribute defines the maximum expected power draw of a load (or the minimum power supplied by a generator) during operation. This value differs from the rated power since it may take into account operational considerations such as environmental, equipment safety or regulatory conditions. |
| adjustedFullDRPower | float | This attribute defines the minimum expected power draw of a load (or the maximum power supplied by a generator) during operation. This value differs from the rated power since it may take into account operational considerations such as environmental, equipment safety or regulatory conditions. |
| powerCurves | PowerCurve | This attribute defines one or more piecewise linear curves mapping the percentage of power consumed by the device as a function of the present level of operation of the device. Many loads draw power (or generators supply power) based on the present loading characteristics of the device. For example, a motor driving a fan will draw more power as the fan blade pitch is increased.  The axes of the curve are defined in percent to allow loads of any type to utilize the attribute. For example a simple 60W incandescent light bulb would likely be modeled by a linear segment (0,0), (100,100). Here as the bulb is energized (dimmed) from off (0%) to fully on (100%), the power needed to energize the device also travels from 0% to 100%. A more complicated device such as a room air-conditioner may have a non-linear power curve. Here, as the coolness setting is adjusted from warmest to coolest, the air conditioner will draw relative more power when set to maximum cooling than when set to minimum cooling.  When powerCurve is not present, the load or generator is assumed to be a two state device drawing no power when the device is off and adjustedNoDRPower when the device is on. When adjustedNoDRPower also is not present, the load or generator is assumed to be a two state device drawing no power when the device is off and maximumRealPower when the device is on. |
| activePowerCurve | int | This attribute defines the index into the zero based array of powerCurves indicating which powerCurve is presently active. |

### PriceCurve

A price curve. Depending on sign of the PriceCurveComponent.quantity it can be price of supply or demand.

| **Name** | **Type** | **Notes** |
| --- | --- | --- |
| priceCurveComponent | PriceCurveComponent | A component of a price curve. |

### PriceCurveComponent

A component of a pricing curve.

| **Name** | **Type** | **Notes** |
| --- | --- | --- |
| price | float | Price of commodity. |
| quantity | float | Quantity of commodity (signed number) can be supply or demand. |
| type | char | Type of commodity -- W, Var, V, Frequency, Wh, .... |

### Quote

| **Name** | **Type** | **Notes** |
| --- | --- | --- |
| quote | TenderComponent | Array of tender components. |

### ResourceControl

Parameters used by local controller to control the resource.

### ResourceLogicalState

### ResourcePhysicalState

ResourcePhysicalState describes physical state parameters for the resource.

| **Name** | **Type** | **Notes** |
| --- | --- | --- |
| power | Power |  |
| phases | Phases |  |

### ResourceStatus

The status of the resource provided describing current load and other conditions.

| **Name** | **Type** | **Notes** |
| --- | --- | --- |
| current | Current |  |
| power | Power |  |
| voltage | Voltage |  |
| phases | Phases |  |
| status | boolean | is resource active. |

### SupervisoryControlSignal

| **Name** | **Type** | **Notes** |
| --- | --- | --- |
| modulationSignal | float | Modulation control signal 0..1.0 for off (0.0) to full load or supply (1.0). |

### Tender

| **Name** | **Type** | **Notes** |
| --- | --- | --- |
| tenderComponent | TenderComponent | Array of time ordered tender components that provides a load or generation profile of price curves. |

### TenderComponent

A component of a tender component.

| **Name** | **Type** | **Notes** |
| --- | --- | --- |
| timeReference | TimeReference | Time reference for this tender component. |
| priceCurve | PriceCurve | Price curve for this time reference. |

### Transaction

| **Name** | **Type** | **Notes** |
| --- | --- | --- |
| accept | boolean | Accept the last quote. |

### Voltage

A complex vector of Voltage and a string enumeration of the phases

| **Name** | **Type** | **Notes** |
| --- | --- | --- |
| voltageByPhase | ComplexNumber |  |

### TimeReference

A time reference, UTC.

### GridNodeId

An integer representing a grid node.

### Phases

A string indicating the list of phases involved -- an order subset of <A,B,C,N,L1,L2>

### StorageType

An enumeration that defines the energy storage characteristics of an instance of the Generator Class.

| **Name** | **Type** | **Notes** |
| --- | --- | --- |
| none |  | This value indicates that this instance of the Generator Class models a device that does not produce energy from storage. |
| electricalStorage |  | This value indicates that this instance of the Generator Class models a device that produces electricity from storage. |
| thermalStorage |  | This value indicates that this instance of the Generator Class models a device that produces thermal energy from storage. |

### SupplyStatusType

This enumeration indicates if the load is presently curtailed and if curtailed is in compliance with the curtailment request received.

| **Name** | **Type** | **Notes** |
| --- | --- | --- |
| supplyInactive |  | This generator is not presently operating |
| supplyRequestPending |  | A request has been received and is pending. |
| supplyCompliant |  | The generator operation is compliant with the last request. |
| supplyNoncompliant |  | The generator is not compliant with the last request. |

### LoadStatusType

This enumeration provides the present overall state of the load.

| **Name** | **Type** | **Notes** |
| --- | --- | --- |
| NA |  | Not applicable |
| eligible |  | The load is presently communicating properly and its data values are correct. In addition, for curtailable loads this load is presently eligible to be curtailed. |
| loadLocked |  | The load is ineligible for curtailment since it has been locked. |
| loadOverridden |  | An external process has set this override attribute prohibiting curtailment. |
| lostCommunication |  | The load presently cannot be accessed. |
| maxCurtailQueue |  | The load is in curtailment and presently being timed for the maximum curtailment time. |
| maxCyclesThisPeriod |  | The load has been cycled the maximum number of times this period. |
| minCurtailQueue |  | The load is in curtailment and presently being timed for the minimum curtailment time. |
| pointNotConsumingEnergy |  | The load is ineligible for curtailment since the point associated with the load is already shut off and not consuming any energy. |
| pointUnreliable |  | The load is ineligible for curtailment since the point associated with the load is unreliable. 'Unreliable' is an error condition when the present value of a point is questioned due to some hardware or software failure. When a point is unreliable, it still may present a value (e.g., Space Temp Present Value = 67 DegF) but carries along a second attribute that indicates this value is suspect. When a point is in the unreliable state, it shall not be curtailed. |
| ramping |  | The load is ramping. That is, it is a transitional state and is either starting up or shutting down. While in this temporary state, it is not eligible for curtailment. |
| releaseQueue |  | The load is ineligible for curtailment. The load has completed its curtailment and is presently timing down the restore time before it is again eligible for curtailment. |
| unlocked |  | The load has recently been unlocked. It will analyze all conditions and set its present eligibility state after analysis completes. |

# Scenario

This section presents the experimental scenario for Transactive Energy simulations. This canonical experiment scenario provides for the orchestration of a transactive energy experiment.

It provides that, for any set of transactive energy components, that are based on the models of this document, a common experiment engine can be run which will produce the results of the simulation that can be compared.

It assumes the following:

1. The platform on which the experiment is running has a class model based on the Abstract Component Model.
2. The interfaces of the model are implemented as publish-subscribe messaging pattern or an equivalent.
3. The interfaces at least provide the minimum data from the component model interface definitions.

Note that whether the model components can be combined from different sources on any given platform can't be guaranteed by the abstract model. But if they are compatible the experiments can be composed.

## Base TE Experiment Scenario diagram

Base scenario for TE experiments. This sequence diagram contains the following:

1. Initialization of all components by the Experiment Manager
2. Three parallel sequences that continue until experiment ends:

* Physical: represents the timing needed to perform multiphysics power and energy simulation.
* Logical Controller: represents the timing needed to perform supervisory and local control of resources.
* Transactive: represents the timing needed to perform a periodic transactive sequence resulting in a pricing model for the duration of the transactive step. This includes a "settle" loop for performing market/participant convergence on price.

This diagram illustrates the data flows and the target destinations of data for those components to use. In implementing this model, pub-sub or request response can produce equivalent results and the arrows and data flowed interpreted appropriately to the message mechanism.

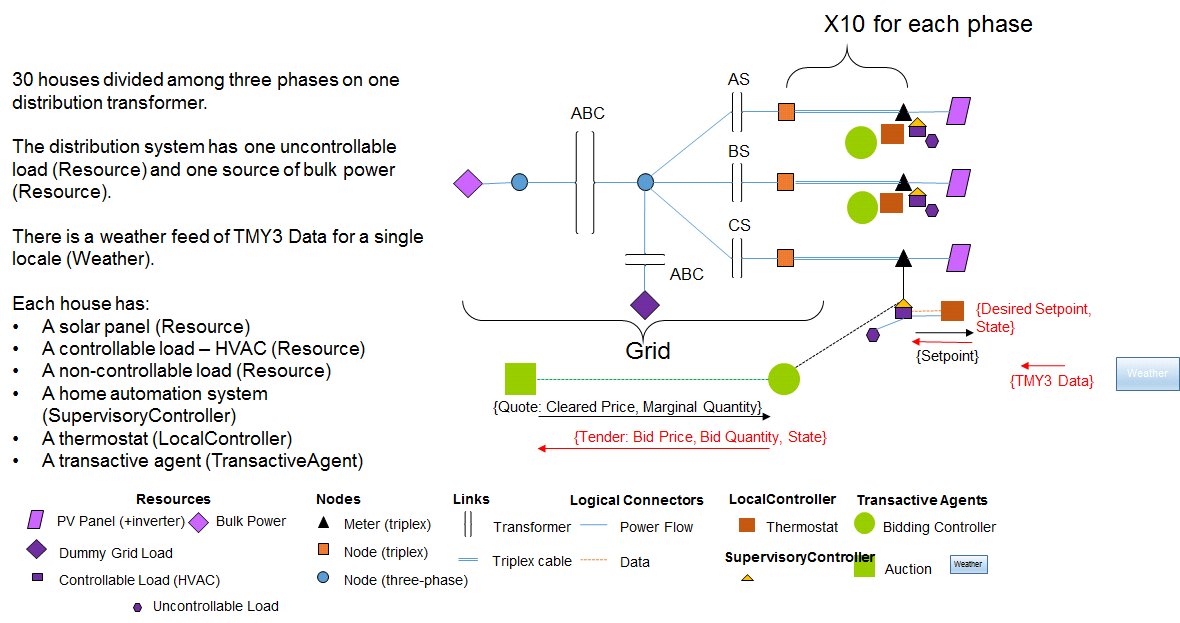


1. Base TE Experiment Scenario

# Beta Use Case

A BetaUseCase was provided by PNNL contributors to this modeling element. The use case itself was realized completely within gridlab-d and is available from: <https://github.com/usnistgov/TEChallengeComponentModel>.

The following figure illustrates the 30 house "beta use case" scenario implemented entirely in Gridlab-D. It presents a minimal distribution segment that exposes three single phase transformers each feeding ten (10) houses. The houses are composed of an HVAC simulation, various loads, and the ability to bid into a transactive energy double-auction model.



1. Beta Use Case

The balance of this section will illustrate the beta use case and its components, including the traceability to the Abstract Component Model.

## Beta Use Case diagram

The Beta Use Case is a scenario chosen to illustrate the interactions of the components of the TE Abstract Component Model. Since its main goal is to illustrate the workings of the TE components and provide a reference simulation to use in developing more realistic and useful use cases, it should be considered in that context.

Once realized on several simulation framework platforms, it provides the basis for comparison and baselining of these platforms.

Shown is a common grid model that supports three sets of 10 house models. One example of each phase-connected house is provided.

The House models, bid into a common Auction. All receive a common set of weather conditions.

Separate diagrams are presented to contain the Gridlab-D models of the Grid, PhaseAHouse, PhaseBHouse, and PhaseCHouse. In each of these diagrams, object instances based on the classes from the Common Component Inheritance diagram are arranged with their initial configuration parameters set.



1. Beta Use Case

## Common Component Inheritance

This package describes the specific classes derived from the Abstract Component Model components and associated additional data types used in the simulation of the Beta Use Case scenario.

### Common Component Inheritance diagram

Shown in the figure is the inheritance model of the Gridlab-D model classes mapped to Abstract Component model.

Note that some parts of the Gridlab-D classes are not distinct but do correspond to the functions of the component model. In these cases, pseudo classes are identified in the diagram to make the inheritance map complete.

Additionally, several Gridlab-D classes are used in composing the Grid and are not represented uniquely in the component model. These are identified in this diagram for reference.



1. Common Component Inheritance

### Auction

| **Name** | **Type** | **Notes** |
| --- | --- | --- |
| current\_price\_stdev\_24h | double |  |
| current\_price\_mean\_24h | double |  |

### BillDump

| **Name** | **Type** | **Notes** |
| --- | --- | --- |
| filename | char |  |
| runtime | char |  |
| group | char |  |

### Climate

### Clock

| **Name** | **Type** | **Notes** |
| --- | --- | --- |
| timezone | char |  |
| starttime | char |  |
| stoptime | char |  |

### GroupRecorder

| **Name** | **Type** | **Notes** |
| --- | --- | --- |
| group | char |  |

### HVAC\_Controller

gridlab-d "controller"

| **Name** | **Type** | **Notes** |
| --- | --- | --- |
| name | char |  |
| market | char |  |
| period | int |  |
| control\_mode | char |  |
| base\_setpoint | char |  |
| setpoint | char |  |
| target | char |  |
| deadband | char |  |
| average\_target | char |  |
| standard\_deviation\_target | char |  |
| demand | char |  |
| total | char |  |
| load | char |  |
| state | char |  |
| use\_predictive\_bidding | boolean |  |
| range\_high | int |  |
| range\_low | int |  |
| ramp\_high | int |  |
| ramp\_low | int |  |

### HVAC\_Load

### House

| **Name** | **Type** | **Notes** |
| --- | --- | --- |
| name | char |  |
| parent | char |  |
| schedule\_skew | int |  |
| Rroof | boolean |  |
| Rwall | double |  |
| Rfloor | double |  |
| Rdoors | int |  |
| Rwindows | double |  |
| airchange\_per\_hour | double |  |
| hvac\_power\_factor | char |  |
| cooling\_system\_type | char |  |
| heating\_system\_type | char |  |
| fan\_type | char |  |
| hvac\_breaker\_rating | int |  |
| total\_thermal\_mass\_per\_floor\_area | int |  |
| motor\_efficiency | char |  |
| motor\_model | char |  |
| cooling\_COP | double |  |
| floor\_area | double |  |
| number\_of\_doors | int |  |
| air\_temperature | double |  |
| mass\_temperature | double |  |
| heating\_setpoint | char |  |

### Inverter

| **Name** | **Type** | **Notes** |
| --- | --- | --- |
| name | char |  |
| inverter\_type | char |  |
| use\_multipoint\_efficiency | boolean |  |
| inverter\_manufacturer | char |  |
| maximum\_dc\_power | int |  |
| four\_guadrant\_control\_mode | char |  |
| generator\_status | GeneratorStatus |  |
| inverter\_efficiency | double |  |
| phase | char |  |
| power\_factor | double |  |
| rated\_power | double |  |
| solar | Solar |  |

### Meter

| **Name** | **Type** | **Notes** |
| --- | --- | --- |
| nominal\_voltage | double |  |
| bustype | char |  |

### Null\_Controller

### Player

| **Name** | **Type** | **Notes** |
| --- | --- | --- |
| name | char |  |
| file | char |  |
| interpolate | char |  |
| loop | int |  |
| value | int |  |

### Recorder

| **Name** | **Type** | **Notes** |
| --- | --- | --- |
| parent | char |  |
| property | char |  |
| interval | int |  |
| file | char |  |

### Solar

| **Name** | **Type** | **Notes** |
| --- | --- | --- |
| panel\_type | char |  |
| orientation | char |  |
| rated\_power | double |  |
| generator\_mode | char |  |
| generator\_status | char |  |

### Substation

| **Name** | **Type** | **Notes** |
| --- | --- | --- |
| name | char |  |
| phases | Phases |  |
| nominal\_voltage | Voltage |  |
| voltage\_A | Voltage |  |
| voltage\_B | Voltage |  |
| voltage\_C | Voltage |  |
| busType | Voltage |  |
| reference\_phase | char |  |

### Transformer

| **Name** | **Type** | **Notes** |
| --- | --- | --- |
| inputs | Phases |  |
| outputs | Phases |  |
| inputId | GridNodeId |  |
| outputId | GridNodeId |  |
| name | char |  |
| groupid | Phases |  |
| phases | Phases |  |
| from | char |  |
| to | char |  |
| configuration | Phases |  |

### Transformer\_configuration

| **Name** | **Type** | **Notes** |
| --- | --- | --- |
| name | char |  |
| connect\_type | char |  |
| install\_type | char |  |
| power\_rating | double |  |
| powerC\_rating | double |  |
| primary\_voltage | double |  |
| secondary\_voltage | double |  |
| impedance | Impedance |  |
| powerA\_rating | double |  |
| powerB\_rating | double |  |

### Triplex\_line

| **Name** | **Type** | **Notes** |
| --- | --- | --- |
| groupId | char |  |
| configuration | char |  |

### Triplex\_line\_conductor

| **Name** | **Type** | **Notes** |
| --- | --- | --- |
| name | char |  |
| resistance | double |  |
| geometric\_mean\_radius | double |  |

### Triplex\_line\_configuration

| **Name** | **Type** | **Notes** |
| --- | --- | --- |
| name | char |  |
| conductor\_1 | char |  |
| conductor\_2 | char |  |
| conductor\_3 | char |  |
| insulation\_thickness | double |  |
| diameter | double |  |

### Triplex\_meter

| **Name** | **Type** | **Notes** |
| --- | --- | --- |
| groupid | char |  |
| parent | char |  |
| price\_base | char |  |
| first\_tier\_price | char |  |
| second\_tier\_price | char |  |
| third\_tier\_price | char |  |
| first\_tier\_energy | double |  |
| second\_tier\_energy | double |  |
| third\_tier\_energy | double |  |
| bill\_day | int |  |
| monthly\_fee | double |  |
| bill\_mode | char |  |
| invertor | Inverter |  |

### Unresp\_load

| **Name** | **Type** | **Notes** |
| --- | --- | --- |
| base\_power\_A | char |  |
| base\_power\_B | char |  |
| base\_power\_C | char |  |
| power\_pf\_A | double |  |
| power\_pf\_B | double |  |
| power\_pf\_C | double |  |
| power\_fraction\_A | double |  |
| power\_fraction\_B | double |  |
| power\_fraction\_C | double |  |
| current\_fraction\_A | double |  |
| current\_fraction\_B | double |  |
| current\_fraction\_C | double |  |
| impedance\_fraction\_A | double |  |
| impedance\_fraction\_B | double |  |
| impedance\_fraction\_C | double |  |
| parent | char |  |
| nominal\_voltage | int |  |

### ZIPLoad

| **Name** | **Type** | **Notes** |
| --- | --- | --- |
| schedule\_skew | int |  |
| heat\_fraction | double |  |
| base\_power | char |  |
| power\_pf | int |  |
| current\_pf | int |  |
| current\_fraction | int |  |
| impedance\_pf | int |  |
| impedance\_fraction | int |  |
| power\_fraction | int |  |

### ZIPLoad\_controller

### GeneratorMode

| **Name** | **Type** | **Notes** |
| --- | --- | --- |
| UNKNOWN |  |  |
| CONSTANT\_V |  |  |
| CONSTANT\_PQ |  |  |
| CONSTANT\_PF |  |  |
| SUPPLY\_DRIVEN |  |  |

### GeneratorStatus

| **Name** | **Type** | **Notes** |
| --- | --- | --- |
| ONLINE |  |  |
| OFFLINE |  |  |

### InverterType

| **Name** | **Type** | **Notes** |
| --- | --- | --- |
| TWO\_PULSE |  |  |
| SIX\_PULSE |  |  |
| TWELVE\_PULSE |  |  |
| OWM |  |  |

## Grid

This package describes the beta use case grid model.

### Grid diagram

This package describes the beta use case grid model.



1. Grid

## PhaseAHouse

This package describes the beta use case model for a house on Phase A. Note that there are 10 such houses and only one is shown.

### PhaseAHouse diagram



1. PhaseAHouse

## PhaseBHouse

This package describes the beta use case model for a house on Phase B. Note that there are 10 such houses and only one is shown.

### PhaseBHouse diagram



1. PhaseBHouse

## PhaseCHouse

This package describes the beta use case model for a house on Phase C. Note that there are 10 such houses and only one is shown.

### PhaseCHouse diagram



1. PhaseCHouse

# Composite and Extended Classes

Composite classes allows for the composition and testing of classes that realize selected sets of interfaces. In any given instance of a TE Component, one or more of the roles or interfaces may be realized. Exchanges shown in the Base TE Experiment Scenario between components that have been combined do not occur during the experiment.

## Composite and Extended Classes diagram

This diagram illustrates how components can be combined and/or extended for use in experiments of the Abstract Component Model.



1. Composite and Extended Classes

## AllInOne

Contains Resource, Local Controller, Supervisory Controller with Transactive interface.

## ExtendedSupervisoryController

This extended SupervisoryController exposes an additional CustomInterface. By substituting this version of the SupervisoryController additional capabilities can be exposed when substituting from the base.

## ResourceWithLocalController

Contains Resource with Local Controller.