01 OPENSTUDIO TODAY

Two schemas and two ways to represent a building model

OpenStudio exposes a typed Model API backed by OpenStudio.idd. The Forward Translator converts that model to an EnergyPlus IDF.

OS MODELING PLATFORM OpenStudio

SCHEMA → OBJECTS

MODEL SCHEMA

OpenStudio.idd

563 OBJECT TYPES

A similar—but independently designed—set of model records

BACKS →

PRIMARY INTERFACE

Model API

```
model = OpenStudio::Model::Model.new
building = model.getBuilding
```

Typed classes, object relationships, and Model helper methods

`.osm` text persists the model

SIMILAR ≠ DIFFERENT

E+ SIMULATION ENGINE EnergyPlus

SCHEMA → TEXT

INPUT SCHEMA

Energy+.idd

858 OBJECT TYPES

Objects, fields, units, choices, defaults, and references

DEFINES →

INPUT MODEL

.idf text

```
Building,
  Example Office,
  0.0,          !- North Axis
  Suburbs;      !- Terrain
```

The text file is the simulation payload.

WORK HERE **OpenStudio Model** → FORWARD TRANSLATOR → GENERATE **EnergyPlus IDF** → RUN **EnergyPlus**

The Forward Translator converts the OpenStudio Model to EnergyPlus IDF.

02 THE PROJECT GOAL

Make the OpenStudio Model work directly on EnergyPlus IDF

Resolve differences in object names, fields, relationships, and behavior while keeping the OpenStudio Model API.

OS

THE FAMILIAR INTERFACE

OpenStudio Model API

PRESERVE

Keep the interface that applications, measures, and users already know.

- Typed objects
- Relationships
- Model helper methods

RESOLVE DIFFERENCES

Schema differences

Implement OpenStudio Model behavior on EnergyPlus IDF objects.

- Names & fields
- Corresponding objects
- Object relationships

E+

THE MODEL SAVED AS IDF

EnergyPlus IDD + IDF

USE DIRECTLY

Load, edit, and save EnergyPlus IDF directly instead of maintaining a second OpenStudio schema.

- Energy+.idd defines the schema
- IDF text stores the model
- Load · edit · save directly

Keep the OpenStudio modeling experience. → Make EnergyPlus IDF the model underneath it.

03 THE EASY CASE

The easy case: the Model class and IDD object line up

Original OpenStudio has a CurveQuadratic class with scalar methods named after the EnergyPlus fields.

OPENSTUDIO MODEL · RUBY

CurveQuadratic

MODEL CLASS

```ruby
curve = OpenStudio::Model::CurveQuadratic.new(model)

curve.setName("Capacity Modifier")

curve.setCoefficient1Constant(0.8)

curve.setCoefficient2x(0.2)

curve.setCoefficient3xPOW2(0.0)

curve.setMinimumValueofx(0.0)

curve.setMaximumValueofx(1.0)
```

NEAR 1:1

ENERGYPLUS IDF

Curve:Quadratic

IDD TYPE

```
Curve:Quadratic,

  Capacity Modifier,  !- Name

  0.8,                !- Coefficient1 Constant

  0.2,                !- Coefficient2 x

  0.0,                !- Coefficient3 x**2

  0.0,                !- Minimum Value of x

  1.0;                !- Maximum Value of x
```

EVEN HERE **OpenStudio still maintains its own schema, its own Model objects, and the translation code that writes EnergyPlus IDF.**

04 SMALL DIFFERENCES APPEAR

Mostly parallel—but the names no longer match exactly

The object still maps cleanly, but type and field names are no longer identical.

OPENSTUDIO MODEL · RUBY

CoilHeating Gas

MODEL CLASS

```ruby
coil = OpenStudio::Model::CoilHeatingGas.new(model)
coil.setName("Main Heating Coil")
coil.setFuelType("NaturalGas")
coil.setGasBurnerEfficiency(0.85)
coil.setNominalCapacity(10000)
coil.setOnCycleParasiticElectricLoad(50)
coil.setOffCycleParasiticGasLoad(100)
```

SAME IDEA

ENERGYPLUS IDF

Coil:Heating: Fuel

IDD TYPE

```
Coil:Heating:Fuel,
  Main Heating Coil,          !- Name
  Always On Discrete,         !- Availability Schedule Name
  NaturalGas,                 !- Fuel Type
  0.85,                       !- Burner Efficiency
  10000,                      !- Nominal Capacity
  Heating Coil Inlet Node,    !- Air Inlet Node Name
  Heating Coil Outlet Node,   !- Air Outlet Node Name
  ,                           !- Temperature Setpoint Node Name
  50,                         !- On Cycle Parasitic Electric Load
  ,                           !- Part Load Fraction Correlation Curve Name
  100;                        !- Off Cycle Parasitic Fuel Load
```

? **Where are the node setters?**
The simple field mapping ends here.

Gas ↔ Fuel | GasBurnerEfficiency ↔ Burner Efficiency | GasLoad ↔ Fuel Load

The scalar fields still map. The node fields hint at something structurally different.

05 THE FIRST STRUCTURAL DIVERGENCE

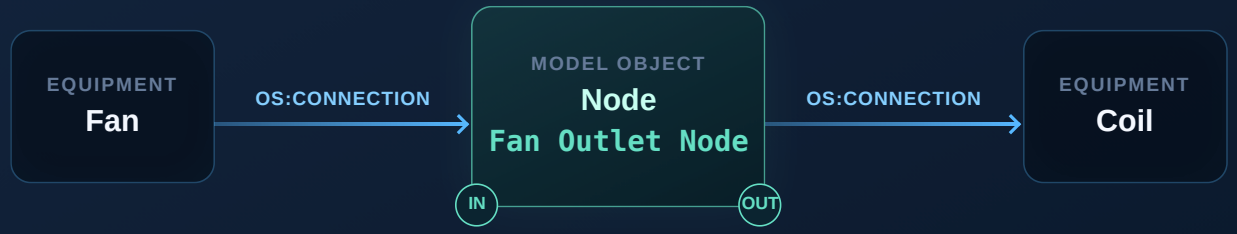
Nodes: objects in OpenStudio, shared names in EnergyPlus

OpenStudio connects equipment through a Node object. EnergyPlus repeats “Fan Outlet Node” in the adjoining equipment fields.

OS

OPENSTUDIO MODEL

Node and Connection objects



vs

E+

ENERGYPLUS IDF

Shared node names

Fan:ConstantVolume

Air Outlet Node Name — `"Fan Outlet Node"`

SAME NAME

Coil:Heating:Fuel

Air Inlet Node Name — `"Fan Outlet Node"`

Aa **Same text means the same node.**
No Node object exists in the IDF.

The OpenStudio Node named “Fan Outlet Node” is a Model object connected to the fan and coil.

In the IDF, “Fan Outlet Node” is simply repeated in the two equipment fields.

06 A SECOND STRUCTURAL DIVERGENCE

OpenStudio groups default schedules; EnergyPlus assigns each one directly

SpaceType and DefaultScheduleSet are OpenStudio Model objects. EnergyPlus has Space, but not SpaceType or DefaultScheduleSet; each load names its schedule directly.

OS

OPENSTUDIO MODEL

Assign the defaults once

OPENSTUDIO SPACETYPE

Office

setDefaultScheduleSet →

OS:DEFAULTSCHEDULESET

Office Defaults

People	Occupancy Schedule
Lights	Lighting Schedule
Equipment	Equipment Schedule

```
space_type.setDefaultScheduleSet(office_defaults)
```

vs

E+

ENERGYPLUS IDF

Write each schedule explicitly

`People`	Schedule Name	Occupancy Schedule
`Lights`	Schedule Name	Lighting Schedule
`ElectricEquipment`	Schedule Name	Equipment Schedule

∅ **No DefaultScheduleSet object**
The schedule assignments remain, but the set does not.

The Forward Translator resolves the defaults. When epmodel opens the IDF, the schedule assignments remain, but the original set is gone. | DefaultConstructionSet has a similar issue.

07 HOW WE RECONCILE THE SCHEMAS

Resolve each difference without changing the Model API

For each difference, we adapt epmodel, propose a change to EnergyPlus, or do both. Existing Measures must continue to run.

ALWAYS Keep the public OpenStudio Model API → The Measure ecosystem keeps working

01

ADAPT EPMODEL

Implement OpenStudio behavior with EnergyPlus objects

EXAMPLE · NODES

Keep Node objects in the Model API

Node methods work as before, while epmodel reads and writes EnergyPlus node names, Branches, and related HVAC objects.

\+

EITHER OR BOTH

02

EVOLVE ENERGYPLUS

Propose a change to the EnergyPlus schema

EXAMPLE · DEFAULTS

Add default schedule sets

Represent the reusable schedule group directly in IDF instead of expanding it into separate load assignments.

FOR EACH DIFFERENCE **Choose the clearest implementation in epmodel, EnergyPlus, or both.** | Then verify existing Model methods and Measures.

08 THE SCALE OF THE EFFORT

Repeating this across the EnergyPlus IDD is a massive task

There are 858 input object types. Each one must be compared with the OpenStudio Model API, implemented, available to Measures, and tested.

ENERGYPLUS 25.2

858

input object types

Each type can bring fields, references, defaults, behavior, bindings, and tests.

Aa

Fields and defaults

Names, types, defaults, autosizing, validation

↔

Object relationships

References, ownership, lists, inheritance

◇

Model behavior

Constructors, helper methods, HVAC connections, removal

✓

Measures and files

Ruby bindings, IDF load and save, focused tests

hundreds of object types × **fields and relationships** × **Model behavior and tests** = **a massive implementation effort**

09 CODING AGENTS ARE INTEGRAL

Coding agents changed how we can take on this work

We started by asking for one change at a time. We now give agents larger, testable assignments and let them work through the IDD in a controlled loop.

STARTED HERE

01

One request at a time

A person chose the next object, supplied the background, watched the work, and reviewed each result.

`“Implement this type.”`

OUR CUSTOM PROGRAM LOOP

02

Keep the work moving

The program records progress, selects the next object, calls an implementation agent, runs reviews, and returns for fixes.

`Plan → Implement → Review ↺`

INCREASINGLY POSSIBLE TODAY

03

Larger assignments

Agents can increasingly plan, implement, test, and correct their work across an object family—or a much larger assignment.

`“Do this across all 800+ types.”`

Coding agents are essential to delivering epmodel. | Their ability to manage long tasks is improving quickly.

10 WHERE WE ARE TODAY

An OpenStudio Measure can already transform an IDF directly

The checked-in example starts with a reference-building IDF, replaces its HVAC systems with familiar Model methods, and produces a simulation-ready IDF.

01 · LOAD

IDF

Existing model

`RefBldgSmallOffice`
`New2004_Chicago.idf`

The OSW names an **.idf** seed—not an OSM.

EPMODEL →

02 · MODIFY

Standard ModelMeasure pattern

```
air_loops = model.getAirLoopHVACs()
air_loops.each { |loop| loop.remove() }

air_loop = OpenStudio::EPModel::AirLoopHVAC.new(model)
air_loop.addBranchForZone(zone, terminal)
```

The Measure uses familiar Model methods to build the new air and plant loops.

SAVE →

03 · RUN

✓

Modified IDF

`model-measures.idf`

EnergyPlus completed successfully.

5 → 1	5	2	0
AIR LOOPS	ZONES REATTACHED	NEW PLANT LOOPS	MEASURE ERRORS

Boiler hot water + air-cooled chiller + multizone VAV

11 THE NEXT MILESTONE

Alpha by the end of the fiscal year

TARGET · FALL 2026

ALPHA

A usable epmodel release for early evaluation and feedback.

Open the presentation

dev-8080.kbenne.net

PDF **Download the slides**

12 BONUS

Could one agent prompt generate this entire deck?

Plausibly. It would look something like this:

AGENT PROMPT

ONE SHOT

Working in this OpenStudio repository, first read the epmodel documentation and inspect the relevant source, IDD files, and checked-in workflow examples.

Build a polished, self-contained 16:9 HTML/CSS/JS presentation for OpenStudio modelers. Explain today's two-schema setup and the goal of making the Model API work directly on EnergyPlus IDF while preserving existing Model APIs and Measures. Keep OpenStudio on the left and EnergyPlus on the right.

Use repository-backed examples to progress from CurveQuadratic, to CoilHeatingGas versus Coil:Heating:Fuel, to Nodes and Connections, to SpaceType and DefaultScheduleSet. Explain the case-by-case reconciliation strategy, the scale across the IDD, why coding agents are integral, the working IDF Measure workflow, and the fall alpha target.

Use concise human language, familiar OpenStudio vocabulary, large projection-safe type, fullscreen keyboard navigation, a QR code, and a linked PDF. Serve it on port 8080. Render every slide and fix any crowding, clipping, small text, or unsupported claims before calling it done.

Plausible today. | The important parts are repository access, a concrete brief, and a careful review of the rendered result.