

BATEM architecture (UML summary)

High-level structure of the `batem` Python package and the main simulation pipeline.
Diagrams use [Mermaid](#) syntax (UML class / component / sequence style).

1. Package map (component view)

```
flowchart TB
  subgraph entry["Entry points"]
    batermCLI["batem.baterm<br/>cmd2 shell"]
    notebooks["Notebooks / scripts<br/>batem.sites"]
    forge["forge<br/>EnergyPlus, utilities"]
  end

  subgraph core["batem.core - building physics & simulation"]
    building["building"]
    data["data"]
    control["control"]
    model["model"]
    thermal["thermal"]
    components["components"]
    solar["solar"]
    library["library"]
    weather_mod["weather_mod"]
    climate["climate"]
    comfort["comfort / inhabitants"]
    enercost["enercost"]
  end

  subgraph apps["Application layers"]
    baterm_mix["batem.baterm.mixins<br/>context, building, pv, -"]
    ecommunity["batem.ecommunity<br/>multi-dwelling managers"]
    graphs_hp["batem.graphs.heatpumps<br/>TESPy cycle models"]
    reno["batem.reno<br/>renovation workflows"]
    tools["batem.tools"]
  end

  subgraph data_assets["Persistent data"]
    xlsx["propertiesDB.xlsx"]
    json_ctx["*.context JSON"]
    json_bld["*.building JSON"]
    json_pv["*.pv JSON"]
    weather_files["Weather archives"]
  end

  batermCLI --> baterm_mix
  baterm_mix --> building
  baterm_mix --> solar
  baterm_mix --> enercost
  notebooks --> building
  notebooks --> core

  building --> data
  building --> control
  building --> model
  building --> solar
  building --> library
  building --> comfort

  model --> thermal
  model --> components
  model --> data
  control --> data
  solar --> data
  data --> weather_mod
  weather_mod --> weather_files
  library --> xlsx

  building -. -> json_ctx
  building -. -> json_bld
  solar -. -> json_pv

  ecommunity --> control
  ecommunity --> data
  graphs_hp --> tools
  reno --> building
```

2. Core simulation class diagram

Configuration dataclasses are edited in GUIs or JSON; **Building** wires subsystems and runs the annual hourly loop.

```

classDiagram
    direction TB
    class ContextData {
        <float latitude_north_deg
        <float longitude_east_deg
        <int location
        <list<SideMaskData> side_masks
        <int simulated_year
    }
    class BuildingData {
        <int footprint
        <int n_floors
        <dict compositions
        <list wall_contacts
        <float hvac_cop_heating
        <...
    }
    class SideMaskData {
        <str name
        <float x_center, y_center
        <float width, height, elevation
        <float exposure_deg, slope_deg
    }
    class PvpPlantData {
        <ContextData context
        <int number_of_panels
        <MOUNT_TYPES mount_type
        <...
    }
    class Context {
        <ContextData context_data
        <DataServiceProvider d
        <list distant_masks
    }
    class Building {
        <Context context
        <BuildingData building_data
        <DataServiceProvider d
        <ModeMaker mode_maker
        <Simulation simulation
        <list zone_floors
        <simulate() BuildingResult
    }
    class DataServiceProvider {
        <WeatherData weather_data
        <Windings bindings
        <list datetimes
        <add_vo(name, series)
        <series(name) list
    }
    class SolarModel {
        <mpoly_mask()
        <best_angles()
    }
    class ModeMaker {
        <dict state_modes_cache
        <StateModel nominal_state_model
        <make(k) StateModel
        <zones_to_simulate()
    }
    class ThermalNetwork {
        <build from Zone, Wall, Side
    }
    class StateModel {
        <discrete-time RC dynamics
        <simulate step
    }
    class Simulation {
        <TemperatureController controllers
        <run annual loop
    }
    class SignalGenerator {
        <OCCUPANCY, MODE, SETPOINT
    }
    class PvpPlant {
        <PvpPlantData data
        <annual production
        <best_angles()
    }
    class Properties {
        <load(short_name, sheet, row)
        <properties@b.xlsx
    }
    ContextData "1" --> "1" Context : builds

```

3. **baterm** interactive shell (composition)

The CLI is a **mixin composition** on `cmd2.Cmd` ; session state holds loaded JSON models.

```
classDiagram
    direction LR

    class Cmd2 {
        <<cmd2.Cmd>>
    }

    class Baterm {
        +ContextData context_data
        +BuildingData building_data
        +Building building
        +PVplantData pv_plant_data
        +list side_masks
        +data_provider DataProvider
    }

    class WorkspaceCommandsMixin
    class ContextCommandsMixin
    class BuildingCommandsMixin
    class MaterialCommandsMixin
    class PvCommandsMixin
    class EnercostCommandsMixin
    class SideMaskCommandsMixin
    class PlottingMixin
    class SummaryMixin

    Cmd2 <|-- Baterm
    WorkspaceCommandsMixin <|-- Baterm
    ContextCommandsMixin <|-- Baterm
    BuildingCommandsMixin <|-- Baterm
    MaterialCommandsMixin <|-- Baterm
    PvCommandsMixin <|-- Baterm
    EnercostCommandsMixin <|-- Baterm
    SideMaskCommandsMixin <|-- Baterm
    PlottingMixin <|-- Baterm
    SummaryMixin <|-- Baterm
```

4. Annual simulation sequence (simplified)

```
sequenceDiagram
    actor User
    participant BT as baterm
    participant B as Building
    participant DP as DataProvider
    participant SG as SignalGenerator
    participant MM as ModelMaker
    participant SM as StateModel
    participant Sim as Simulation

    User->>BT: building simulate
    BT->>B: simulate()
    B->>DP: weather, bindings, zone vars
    B->>SG: OCCUPANCY, SETPOINT, MODE per zone
    B->>MM: build thermal network + nominal model
    loop each hour k
        Sim->>MM: make_k() if fingerprint changed
        MM-->>SM: StateModel
        Sim->>SM: state update
        Sim->>DP: PHVAC, TZ, GAIN, OCCUP_ELEC, ...
    end
    B->>DP: HVAC_ELEC, PELEC, PZ:building#sim
    B-->>BT: BuildingResult
```

5. Time-series naming hub

All modules share one **DataProvider** per site/year. Variables follow

`doc/naming_convention.md` (`TZ:floor0` , `PHVAC:floor0#sim` , `PELEC:building#sim` , `PV:power_W` , ...).

```
flowchart LR
    subgraph inputs
        W["weather → TZ:outdoor"]
        SG2["SignalGenerator → OCCUPANCY, SETPOINT, MODE"]
        SOL["SolarModel → irradiance, GAIN_SOLAR, PSOLW"]
    end

    subgraph state
        MM2["ModelMaker / StateModel"]
        CTRL["TemperatureController → PHVAC"]
    end

    subgraph outputs
        TZ2["TZ:zone#sim"]
        ELEC["HVAC_ELEC, PELEC, OCCUP_ELEC"]
        PV2["PV:power_W"]
        EC["ENERCOST, GRID_IMPORT (enercost)"]
    end

    W --> MM2
    SG2 --> MM2
```

6. Related packages (out of core path)

Package	Role
<code>batem.ecommunity</code>	Several dwellings, managers (<code>Basic</code> , <code>Reactive</code> , <code>Dynamic</code> , ...), flexibility
<code>batem.graphs.heatpumps</code>	TESPy heat-pump cycle graphs and dashboards
<code>batem.reno</code>	Renovation / house upgrade workflows
<code>batem.lambdahouse</code> (via <code>core.lambdahouse</code>)	Simplified house model & reporting
<code>batem.weather</code>	Weather file I/O helpers
<code>forge/</code>	EnergyPlus and other external-tool bridges (not imported by core by default)

Viewing the diagrams

- **VS Code / Cursor:** Markdown preview with Mermaid support.
- **GitLab:** Mermaid renders in `.md` files.
- **Export:** [Mermaid Live Editor](#) or `mmdc` (mermaid-cli) for PNG/SVG.

For field-level variable rules, see `doc/naming_convention.md`. For CLI workflows, see `doc/Baterm_User_Manual.md`.