

# BATEM Core API Documentation

**Documentation note:** All modules within `batem.core` expose reStructuredText (RST) docstrings with `.. module::` directives and Sphinx field lists, making them compatible with Sphinx autodoc. Refer to the source files for authoritative parameter descriptions.

## Table of Contents

[TOC]

# Building Modeling

## BuildingData

**Location:** `batem.core.building`

**Purpose:** Captures the geometric, material, ventilation, and occupant assumptions used to assemble a full building model.

### Key Parameters:

- `footprint`, `side_glazing_ratios`, `n_floors`, `floor_height`, `base_elevation`, `z_rotation_angle_deg`
- `glazing_solar_factor`, `max_unshaded_protections` (list of `MaxUnshadedProtection`: `altitude_max_deg`, `azimuth_half_width_deg`)
- `compositions`: envelope layer definitions (`inout_wall`, `inout_roof`, `inout_glazing`, ...)

# Thermal Analysis

## ThermalNetworkMaker

**Location:** `batem.core.thermal`

**Purpose:** Creates thermal networks from building components.

### Constructor

```
ThermalNetworkMaker(*zone_names: str, periodic_depth_seconds: float = 3600)
```

### Parameters:

- `zone_names` : Names of thermal zones
- `periodic_depth_seconds` : Periodic depth for thermal analysis

### Key Methods

# System Components

## Refrigerant

**Location:** `batem.core.system`

**Purpose:** Models refrigerant properties and thermodynamic cycles.

## Constructor

```
Refrigerant(refrigerant_name: str)
```

## Parameters:

- `refrigerant_name`: Name of refrigerant (e.g., "R134a", "R410A")

## Key Methods

```
heat_pump_cycle(T1_cold_K: float, T3_hot_K: float,  
pressure_drop_evaporator_Pa: float = 10000
```

# Data Management

## DataProvider

**Location:** `batem.core.data`

**Purpose:** Main interface for data management in building energy modeling.

## Constructor

```
DataProvider(parameter_set: ParameterSet = None)
```

## Parameters:

- `parameter_set` : ParameterSet instance (optional)

## Key Methods

```
add_variable(name: str, data: numpy.ndarray, time_stamps:  
numpy.ndarray = None)
```

# BATEM terminal ( `baterm.baterm` )

Entry points: `baterm` (console script) or `python -m baterm.baterm`

Documentation: [Baterm\\_User\\_Manual.md](#) · `baterm/baterm/README.md` · in-shell `help`  
`baterm`

Session types: `ContextData` , `BuildingData` , `PVplantData` → shared hourly  
`DataProvider`

| Command family        | Module                                          |
|-----------------------|-------------------------------------------------|
| <code>context</code>  | <code>baterm.baterm.mixins.context_cmds</code>  |
| <code>building</code> | <code>baterm.baterm.mixins.building_cmds</code> |
| <code>pv</code>       | <code>baterm.baterm.mixins.pv_cmds</code>       |
| <code>sidemask</code> | <code>baterm.baterm.mixins.sidemask_cmds</code> |
|                       |                                                 |

# ParameterSet

**Location:** `batem.core.data`

**Purpose:** Manages parameter sets for building models.

## Constructor

```
ParameterSet(name: str = "default")
```

## Parameters:

- `name` : Parameter set name

## Key Methods

```
add_parameter(name: str, value: float, min_value: float =  
None, max_value: float = None, unit: str = None)
```

Adds a parameter to the set.

# Occupant Comfort

## Preference

**Location:** `batem.core.inhabitants`

**Purpose:** Comprehensive occupant preference model for comfort and cost assessment. Quantifies thermal comfort, air-quality comfort, configuration effort, and energy cost from an occupant-centric perspective.

## Constructor:

```
Preference(  
    preferred_temperatures: tuple[float, float] = (20, 26),  
    extreme_temperatures: tuple[float, float] = (16, 30),  
    preferred_CO2_concentration: tuple[float, float] = (500, 1700),  
    temperature_weight_wrt_CO2: float = 0.5,  
    power_weight_wrt_comfort: float = 0.5,  
    sleeping_hours: Iterable[int] | None = (23, 0, 1, 2, 3, 4, 5, 6),  
    mode_cop: dict[int, float] = {}  
)
```



# Control Systems

## ModelPredictiveController

**Location:** `batem.core.control`

**Purpose:** Implements Model Predictive Control for building energy management.

## Constructor

```
ModelPredictiveController(state_model: StateModel, prediction_horizon: int = 24, control_horizon: int = 24)
```

## Parameters:

- `state_model` : State space model
- `prediction_horizon` : Prediction horizon in time steps
- `control_horizon` : Control horizon in time steps

## Key Methods

# Weather Processing

## WeatherData

**Location:** `batem.core.weather`

**Purpose:** Processes and manages weather data.

## Constructor

```
WeatherData(data_source: str = "default")
```

## Parameters:

- `data_source` : Data source identifier

## Key Methods

```
load_data(file_path: str, format: str = "csv")
```

# Utilities

## Properties

**Location:** `batem.core.library`

**Purpose:** Provides material properties and constants.

## Key Methods

**`get(material_name: str) -> dict`**

Gets material properties.

**Parameters:**

- `material_name` : Material name

**Returns:** Dictionary with material properties

**`get_all_properties() -> dict`**

# Error Handling

## Common Exceptions

### **ValueError**

Raised when invalid parameters are provided.

### **RuntimeError**

Raised when system constraints are violated.

### **FileNotFoundError**

Raised when data files cannot be found.

### **ConvergenceError**

Raised when numerical algorithms fail to converge.

# Performance Optimization

## Memory Management

- Use `numpy` arrays for large datasets
- Implement data streaming for large files
- Use sparse matrices for sparse systems
- Clean up unused variables

## Computational Efficiency

- Vectorize operations where possible
- Use appropriate numerical methods
- Implement caching for repeated calculations
- Use parallel processing for independent operations

*This API documentation provides detailed information about the classes and methods in the BATEM core module. For implementation examples and advanced usage, refer to the test files and example notebooks.*