

RESEARCH ARTICLE

CityLearn v3: A Configurable Simulation and Evaluation Framework for Realistic Control Studies of Renewable Energy CommunitiesTiago Fonseca^a, Luis Lino Ferreira^a, Armando Sousa^b, Ava Mohammadi^c and Zoltan Nagy^c^aINESC TEC / Polytechnic of Porto - School of Engineering, Porto, Portugal; ^bFEUP - Faculty of Engineering, University of Porto / INESC TEC, Porto, Portugal; ^cDepartment of the Built Environment, Building Services, Eindhoven University of Technology, Eindhoven, the Netherlands**ABSTRACT**

Renewable energy communities (RECs) coordinate buildings, photovoltaic generation, batteries, electric vehicles and flexible loads. Controller studies often simplify changing participation, equipment availability, service deadlines and data quality, so lower cost or peak demand can conceal missed services or infeasible power requests. This paper presents CityLearn v3, a configurable simulation and evaluation framework for REC control studies under these conditions. It represents changing members and assets, flexible-load deadlines, demand-response requests, local energy sharing, and data or equipment failures within one simulation environment. Building and phase power limits constrain controllable requests, while a declared timestep preserves consistent power-to-energy accounting. The framework records controller inputs and distinguishes requested actions from those applied to the simulated equipment. Reference controllers, service- and constraint-aware performance indicators, and trajectory exports support comparisons within and across communities. Software checks and application examples examine service delivery, electrical constraints, settlement and changing scenarios; a synthetic high-frequency trace replay illustrates how aggregation can conceal short peaks without changing annual energy. Together, these records allow aggregate performance to be interpreted alongside service failures, action reductions and participant-level outcomes.

KEYWORDS

CityLearn; renewable energy communities; building performance simulation; distributed energy resources; electric vehicles; demand response; reinforcement learning; key performance indicators

1. Introduction

Buildings increasingly combine photovoltaic (PV) generation, batteries, heat pumps, flexible appliances and electric-vehicle (EV) chargers. Households, public buildings and businesses can coordinate these resources within a renewable energy community to increase local PV use, reduce peak grid import, provide flexibility and share locally generated energy.

Consider a simple day as an example. In the morning, a new household joins the community and its PV system becomes part of the local energy balance. During the afternoon, an EV arrives at a charger and must reach a required state of charge before departure. A washing-machine or process cycle may start later, but it still has to finish before its deadline. In the evening, the grid operator may ask the community to reduce import for one hour. During the same period, a meter may stop reporting and one battery command may fail. A realistic controller study must state all these conditions and preserve them in the reported results.

Established district-control benchmarks provide shared scenarios and evaluation protocols (Vázquez-Canteli et al. 2020; Nweye et al. 2025). Studies using fixed membership and complete data can, however, leave the changing conditions described above outside the control problem. Established simulation tools cover different parts of the energy-system problem. EnergyPlus and the Modelica Buildings library provide detailed models of buildings, thermal systems and HVAC

equipment; BOPTEST provides standardized test cases, an application programming interface and performance indicators for building-control benchmarking; and HELICS and mosaik connect models from different domains in a common co-simulation (Crawley et al. 2001; Wetter et al. 2014; Blum et al. 2021; Hardy et al. 2024; Steinbrink et al. 2019). CityLearn addresses a different need as it provides a common Gym-compatible environment in which control policies can be tested over the same energy scenarios, controller interfaces and performance indicators. The original CityLearn established this benchmark role (Vázquez-Canteli et al. 2020), and CityLearn v2 extended it with rule-based control (RBC), model-predictive control (MPC), reinforcement-learning control (RLC), distributed energy resources, EV and vehicle-to-grid examples, occupant feedback and resilience studies (Nweye et al. 2025; Towers et al. 2024; Fonseca et al. 2025). Table 1 summarizes these complementary approaches in terms of their modelling scope, control-study interfaces, and representation of community services and constraints.

Table 1.: Base capabilities of selected simulation and control-benchmarking platforms.

Platform	Primary purpose	Control-study interface	Community and service scope
EnergyPlus and Modelica Buildings (Crawley et al. 2001; Wetter et al. 2014)	Detailed building, HVAC and district-energy physics	Control interfaces and component models are available; the experiment protocol is assembled by the user	Community membership, local settlement and common controller-evaluation semantics are study-specific
BOPTEST (Blum et al. 2021)	Repeatable benchmarking of building control using high-fidelity emulators	Standard API, test cases, baseline control and KPIs	Centred on building and HVAC control; REC services and community electrical constraints require additional modelling
HELICS and mosaik (Hardy et al. 2024; Steinbrink et al. 2019)	Synchronization and data exchange across heterogeneous simulators	Coordinates user-supplied models and controllers; evaluation logic is defined by the federation or scenario	Can support multi-domain REC studies, but REC, service and audit semantics are supplied by the coupled components
CityLearn v2 (Nweye et al. 2025)	Data-driven district control benchmarking	Common Gym-compatible loop with reference and learning controllers and shared KPIs	Integrated buildings, distributed energy resources and EV studies within a shared district-control benchmark

These approaches provide complementary foundations for modelling energy systems and benchmarking control. For the REC studies considered here, community membership, service obligations and electrical constraints must also remain consistent from scenario configuration through controller interaction to evaluation. To support such studies, this paper presents CityLearn v3, a configurable simulation and evaluation framework informed by real-community projects. Its technical contribution is an integrated experimental interface that distinguishes requested from applied actions, enforces building, phase and equipment power limits, and preserves power-to-energy consistency across declared timesteps. Changing participants and assets, flexible services, local settlement and data or equipment failures can therefore be studied under a common control and evaluation protocol.

The distinction between requested and applied actions makes controller decisions auditable. A request to charge a disconnected EV remains recorded even though the applied charging power is zero; a battery request reduced by a phase limit is likewise retained alongside the applied value. Service indicators complement these records, so a reduction in cost can be assessed together with missed EV departures or flexible-load deadlines.

Section 2 derives the study requirements, and Section 3 describes their representation. Section 4 sets out controller interfaces, reference policies, performance indicators, software checks and computational costs. Section 5 demonstrates their use in REC experiments, and Section 6 discusses the findings.

2. Requirements for realistic renewable energy community studies

Experience developing and testing REC controllers in the OPEVA and DEMFLEX projects motivates two groups of requirements: the conditions a scenario must represent, and the information the simulator must expose for fair controller comparison.

2.1. *From real communities to simulator requirements*

Community composition, equipment, service obligations and sharing rules affect both the control problem and the interpretation of its results. A scenario must specify these assumptions together with its time resolution, electrical limits and data availability.

Recent REC studies point to the same need. Energy communities are socio-technical systems, so their performance depends on participant composition, technical design, local objectives and governance as well as total demand (Gjorgievski, Cundeva, and Georghiou 2021; European Parliament and Council of the European Union 2018, 2019). Modelling results change with participant types, technologies, demand profiles and energy-sharing rules (Belmar, Baptista, and Neves 2023). Reviews of machine-learning applications likewise argue that data-driven control should reflect how communities are operated, not only abstract load-shifting objectives (Hernandez-Matheus et al. 2022). Simulation studies combine buildings, PV, batteries, grid interaction, data availability and uncertainty (Mousavi Motlagh et al. 2023); use 15-minute decisions, EV availability, trips, flexible-load windows, export limits, local exchange and forecast uncertainty (Frieß et al. 2023); and show that allocation and settlement rules change how community benefits are distributed (Mello et al. 2024).

The same requirements appeared in the authors’ use of CityLearn and EVLearn in EV and energy-community projects. The OPEVA/EnergAIze demonstrator (Horizon Europe grant 101097267) considered a REC with nine houses, an office building, a factory, more than 20 EV chargers, PV generation, batteries and different building loads (SoftCPS Laboratory 2025; OPEVA Consortium 2026; European Commission, CORDIS 2025; Fonseca et al. 2025). Work towards control deployment in that setting made several assumptions concrete. Chargers, meters and PV systems produced data at different frequencies; values could be missing or noisy; commands could fail; measured PV and load traces required consistent energy accounting across timesteps; appliance and process cycles had start windows and completion deadlines; homes had contracted power and phase limits; local energy sharing changed participant costs; and REC members or equipment could enter or leave during the study.

DEMFLEX (COMPETE2030-FEDER-01657200-19215) added the need to compare controllers across communities, represent demand-response requests explicitly, provide structured inputs to advanced controllers and export trajectories and performance indicators that can be inspected outside the simulator (SoftCPS Laboratory 2026; Eurogia2030 2026). These projects motivate the requirements below.

2.2. *Scenario and evaluation requirements*

Table 2 summarizes the scenario requirements, developed in Section 3.

Table 2.: Requirements for realistic REC scenarios.

ID	REC requirement	Real example	How CityLearn v3 represents it
D1	Multiple communities	A project may study several communities but must retain the result of each one.	Synchronized community simulations with local and combined results.
D2	Demand-response requests	A grid operator requests a power change during a stated time window.	Time-stamped interval DR specifications, delivered response, shortfall and payment.

ID	REC requirement	Real example	How CityLearn v3 represents it
D3	Local energy sharing and settlement	PV surplus can meet another member's demand and change both members' costs.	Participant-level local import, local export, and settlement with the grid and community.
D4	Changing community members and assets	A household joins, a charger is installed or a PV system is removed during the study.	Member and equipment changes that update controller inputs, actions and results.
D5	Flexible-load schedules and deadlines	An appliance or process may start operation later, but must complete before a deadline.	Cycle profiles, allowed start windows, deadlines and service indicators.
D6	Data and equipment failures	A meter value is missing, a forecast is biased, a command is lost or a battery is unavailable.	Repeatable changes to measurements, forecasts, commands and equipment availability.
D7	Time resolution and energy units	Data may arrive every 15 s, 15 min or hour, while power and energy must remain distinct.	A declared timestep and conversion checks.
D8	Building and phase power limits	A home has contracted import/export limits and equipment connected to specific phases.	Building-level and per-phase limits applied to controllable power requests.

Table 3 summarizes the evaluation requirements: controller inputs and actions, performance indicators, reference policies, software validation and computational cost. Section 4 describes how these are addressed.

Table 3.: Requirements for fair and reproducible controller comparisons.

ID	Evaluation requirement	Why it matters	How CityLearn v3 addresses it
S1	Controller inputs and applied actions	The user must know what the controller received, requested and actually caused.	Structured observations with persistent identities, availability masks, feasible action ranges and separate requested/applied actions.
S2	Performance indicators and result exports	One reward or aggregate curve can hide service failures and local outcomes.	Extended KPIs with energy, cost, emissions, service and constraint indicators with time-series and comparison exports.
S3	Reference controllers and fair comparisons	A normalized result is meaningful only when the comparison case uses the same given configuration and accounting rules.	Business-as-usual (BAU) and rule-based reference controllers evaluated in the same scenario.
S4	Software validation	More scenario rules create more opportunities for plausible but incorrect results.	Regression tests, physical checks, interface checks, indicator-consistency checks and saved evidence.
S5	Computational cost	High-frequency data, structured inputs and several communities can slow repeated controller studies.	Measurements of run time, memory usage, data export time, data format.

3. Representing realistic REC scenarios in CityLearn v3

The scenario represents the eight REC requirements in Table 2. The following subsections describe their physical or service meaning, configuration and recorded outcomes.

Figure 1 depicts the main simulator modules. Each blue dashed area is one REC; the yellow area shows that several RECs can be run together without losing their individual results. Within each REC, members exchange locally generated energy before the remaining import or export is accounted for at the grid. The arrows from the Grid/DSO-TSO represent demand-response requests, and the hexagonal controller boxes show the information and actions exchanged at each simulation step.

The examples inside the RECs illustrate membership changes, charger installation, flexible-load deadlines and per-phase limits. The box labelled *robustness events* represents missing data,

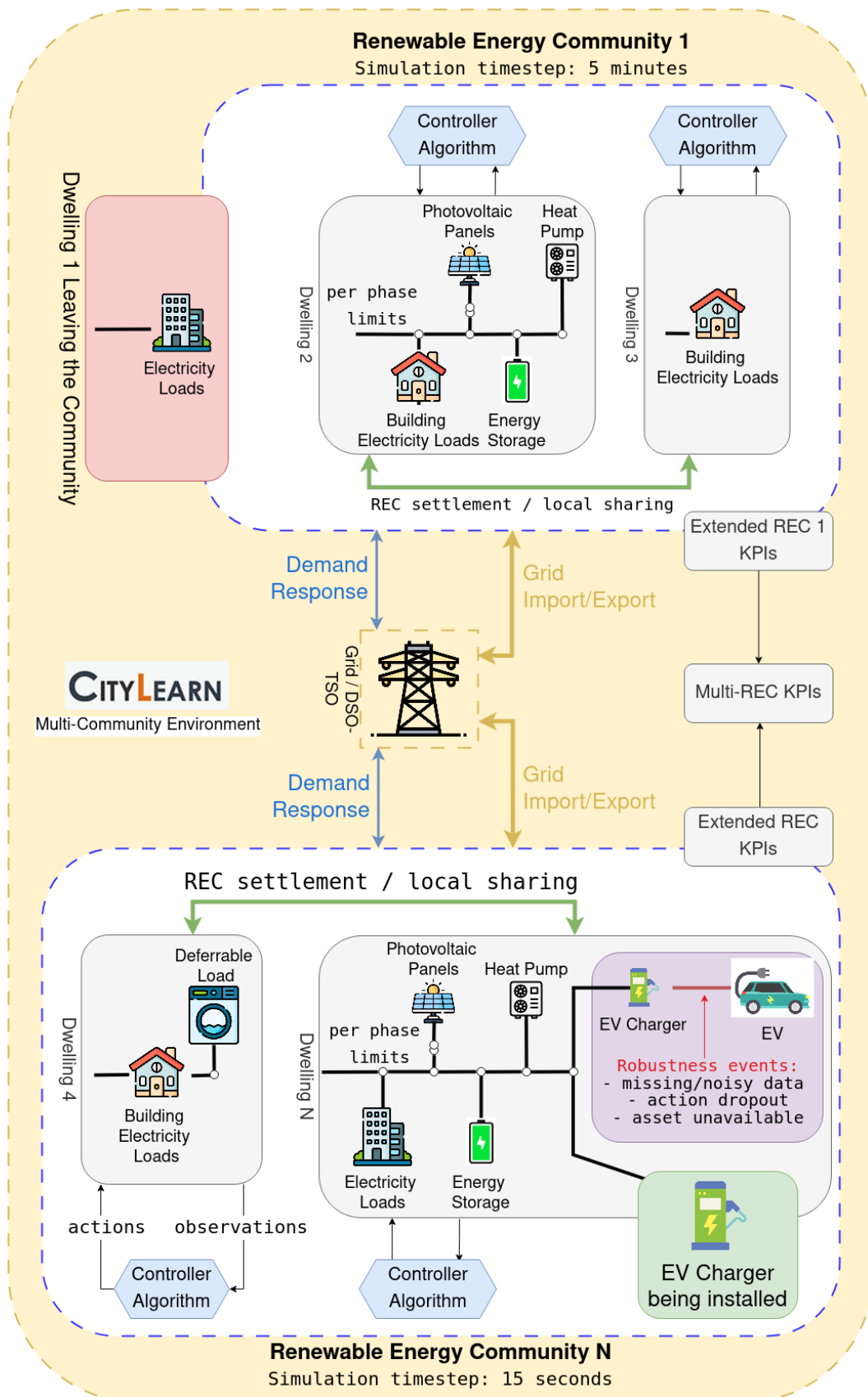


Figure 1.: How CityLearn v3 represents one or more renewable energy communities.

lost commands and unavailable equipment. Appendix A, Table A2, lists the corresponding configuration fields.

3.1. *Multiple communities*

Requirement D1 is the ability to study several communities in one synchronized experiment. A municipality or aggregator may coordinate RECs with different buildings, resources and objectives. A policy can receive observations and provide actions for all communities at the same timestep, while their separate identities and results reveal which community caused a peak, produced surplus or missed a service. The wrapper synchronizes electrically independent community simulations and retains their individual and combined results.

CityLearn v3 runs community environments side by side. Each retains its scenario, members, equipment, demand-response requests, settlement rules and results. The wrapper resets them together and advances each once per simulation step. It accepts actions organized by community identifier and returns observations, rewards and information under those identifiers. The D1 row of Table A2 lists the configuration fields.

The communities evaluated together must use the same centralized or decentralized setting, timestep and episode length. Individual scenarios can use other time resolutions when evaluated separately, but they must be aligned to the same simulation clock before they are included in one multi-community experiment.

Results are reported for each community and for the portfolio. Energy, cost, emissions, payments and event counts are summed; ratios and percentages use declared reporting weights, equal by default. Thus, self-consumption values of 40% and 60% give a mean of 50%, or 55% with weights of 1 and 3. Weights can reflect floor area, member count or annual demand and define averages of community-level ratios.

3.2. *Demand-response requests*

Requirement D2 concerns explicit demand-response requests (Vázquez-Canteli and Nagy 2019). A distribution system operator (DSO), transmission system operator (TSO), retailer or aggregator may ask a community to change its net consumption during a stated interval. The request and its delivery must be visible to the controller and in the results, rather than represented only through a tariff or reward.

Each request specifies its identifier, issuer, start and end timesteps, direction q_r , target power R_r , tolerance ϵ_r , payment rate and shortfall penalty. Delivery is measured against a reference baseline $B_{r,t}$: mean community net power over a configurable pre-event window, one hour by default, calculated immediately before activation and held fixed during the event. This pre-event reference is used consistently for response measurement and settlement. Invalid or incomplete history invalidates the baseline and excludes the request from settlement.

With positive P_t denoting net import, a *down* request asks the community to reduce import relative to the baseline, whereas an *up* request asks it to increase net consumption by increasing import or reducing export. For an active request r , with direction q_r and target power R_r , the delivered power is

$$d_{r,t} = \begin{cases} P_t - B_{r,t}, & q_r = \text{up}, \\ B_{r,t} - P_t, & q_r = \text{down}, \end{cases} \quad (1)$$

Here P_t is actual community net power, positive for import. A positive $d_{r,t}$ indicates movement in the requested direction; a negative value indicates the opposite.

The simulator derives credited power $c_{r,t}$ and shortfall power $s_{r,t}$ from the delivered power. Credited power is the non-negative response that counts towards payment and is capped at the

target R_r . Shortfall is the part still missing after the tolerance ϵ_r :

$$c_{r,t} = \min(\max(d_{r,t}, 0), R_r), \quad s_{r,t} = \max(R_r - d_{r,t} - \epsilon_r, 0), \quad (2)$$

Power is converted to energy by multiplying by the timestep duration in hours. For example, a 20 kW down request during one 15 min step asks for 5 kW h. If import falls by only 14 kW and the tolerance is 1 kW, the credited response is 3.5 kW h and the shortfall is 1.25 kW h. If import falls by 23 kW, the full delivery remains visible, but credit is capped at the 20 kW target and shortfall is zero.

A scenario is defined by a top-level configuration file, normally `schema.json`. Its `demand_response` block enables the service, selects the baseline window and points to a CSV or Parquet request file containing one row per request. The current schema accepts DSO or TSO issuers and non-overlapping requests. During an active request, the structured input includes its identifier, issuer, direction, target power and baseline information. Appendix A, Table A2, gives the exact fields.

3.3. Local energy sharing and settlement

Requirement D3 concerns local energy sharing and settlement. In European REC and citizen-energy-community policy, participation involves collective activity and member or local benefit, not only independent behind-the-meter operation (European Parliament and Council of the European Union 2018, 2019). A member with PV surplus may allocate part of that energy to a neighbour before the remaining export is accounted for at the grid. Batteries, EV charging and flexible loads can then be scheduled to coincide with local surplus. The simulation must retain both the physical allocation and its effect on each member's cost.

District net load cannot identify who supplied or consumed shared energy, or how savings were distributed. For example, simultaneous office PV export and household import can cancel in the aggregate whether or not local sharing is enabled. CityLearn v3 therefore records participant net consumption, local exchange, settlement and residual grid import/export separately.

Let $E_{b,t}^{\text{net}}$ be the net energy of building b during timestep t after local generation and controllable equipment. Positive values are demand and negative values are surplus. For the active members $\mathcal{C}_t$, total demand $D_{\mathcal{C},t}$ and total surplus $S_{\mathcal{C},t}$ are

$$D_{\mathcal{C},t} = \sum_{b \in \mathcal{C}_t} \max(E_{b,t}^{\text{net}}, 0), \quad S_{\mathcal{C},t} = \sum_{b \in \mathcal{C}_t} \max(-E_{b,t}^{\text{net}}, 0). \quad (3)$$

When local sharing is enabled, the shared energy is the smaller of demand and surplus:

$$L_{\mathcal{C},t} = \min(D_{\mathcal{C},t}, S_{\mathcal{C},t}); \quad (4)$$

if sharing is disabled, $L_{\mathcal{C},t} = 0$. Participant weights decide how $L_{\mathcal{C},t}$ is distributed among the members that are importing energy without exceeding their demand. Exporting members contribute in proportion to their surplus. Residual grid import $G_{\mathcal{C},t}^+$ is total demand minus shared energy, and residual grid export $G_{\mathcal{C},t}^-$ is total surplus minus shared energy. The allocation and price settings determine the resulting settlement. As a simple example, suppose that one member has 10 kW h of PV surplus during a timestep and two other members together demand 6 kW h. If all members are eligible for local sharing, the community layer records 6 kW h of local traded energy and 4 kW h of residual export. If a controller shifts an additional appliance or EV charging session into that same period, local traded energy can increase and residual grid exchange can fall. Participant-level records show who supplied and consumed the shared energy and how the configured rule affected costs.

Settlement is calculated from this local energy record rather than inferred later from aggregate net load. The member-level local price $p_{b,t}^{\text{loc}}$ is the retail import price $p_{b,t}^{\text{buy}}$ multiplied by

the configured ratio ρ , where $0 \leq \rho \leq 1$. A value below one prices locally shared energy below retail import. For local import $L_{b,t}^{\text{in}}$, local export $L_{b,t}^{\text{out}}$ and residual grid import $G_{b,t}^+$, the current settlement cost is

$$C_{b,t} = p_{b,t}^{\text{buy}} G_{b,t}^+ + p_{b,t}^{\text{loc}} L_{b,t}^{\text{in}} - p_{b,t}^{\text{loc}} L_{b,t}^{\text{out}}. \quad (5)$$

Imports are costs and local exports are credits. Residual grid export $G_{b,t}^-$ remains in the physical record but has zero remuneration in the current implementation, so it does not appear in this cost equation. Total local import and local export are both equal to the shared energy in Equation (4). The monetary record is budget-balanced when matched energy is charged and credited at one common local price. With different retail tariffs, local credits and charges follow each member's derived local price; a common local price gives exact bilateral balance.

Continuing the example above, if the retail import price is 0.25 EUR/kWh and $\rho = 0.8$, local energy is settled at 0.20 EUR/kWh. A member importing 6 kWh locally pays 1.20 EUR instead of 1.50 EUR from the grid, while the producing member receives a local credit for the energy used by the community. The exported evidence therefore contains both the physical energy allocation and the economic effect: local import, local export, residual grid import/export, settled cost, counterfactual cost, local-market savings and participant/community indicators.

The `community_market` block enables sharing, sets the local-price ratio and optionally supplies import-member weights. These weights allocate scarce surplus among importing members; omitted weights give equal default eligibility. The D3 row of Table A2 lists the fields, exported indicators and legacy price-ratio alias.

3.4. *Changing community members and assets*

Requirement D4 concerns members and equipment that enter or leave during a study. A household may join, a charger may be installed, or a PV system may be removed for maintenance. Representing these changes within one run lets the controller experience the transition while preserving consistent inputs, actions and accounting across periods.

CityLearn v3 stores each change as a time-stamped record with an operation (add or remove), the affected member or asset identifier and, when needed, replacement parameters such as input files, installed power or phase connection. All changes scheduled for timestep t are applied before the controller acts at that timestep. Demand, generation, available actions, power limits, local settlement and service checks therefore use the same active member and asset lists.

In the scenario file, this behaviour is activated with `topology_mode="dynamic"` and a list of `topology_events`. The complete schema stores the events as validated records, but conceptually the declaration is close to:

```
t=1000: add member M4 with demand and PV traces
t=1300: add EV charger to M2
t=1600: remove PV from M1
t=2000: remove member M3
```

Add records can reuse an existing member or item of equipment as a template and replace selected parameters. Remove records close the corresponding active period. These changes are declared explicitly rather than hidden as zero-filled time series or undocumented preprocessing.

Performance indicators follow the active period of each member and asset. A charger installed at $t = 1300$, for example, has no service denominator or action before that timestep. A removed member keeps its historical rows but does not contribute to later settlement. The exported change log explains why member counts, service denominators, local sharing or power-limit results changed during the episode.

3.5. Flexible-load schedules and deadlines

Requirement D5 concerns flexible loads that must deliver a complete service, such as a dishwasher program, water-heating cycle, pump schedule or industrial batch. The controller chooses when the predefined cycle starts, and an omitted cycle is recorded as an unserved service.

In the implementation, these are configured as **deferrable_appliances** using two linked files. The cycle file gives the energy profile $q_{c,\tau}$ in kW h/step, its duration d_c in integer timesteps and total energy. The schedule file gives the earliest start t_c^{earliest} , latest start t_c^{latest} , completion deadline t_c^{deadline} , priority and whether the cycle is mandatory. The controller selects the start time of the stored energy profile.

For a requested cycle c , an applied start s_c is feasible only if

$$s_c \in \mathbb{Z}, \quad t_c^{\text{earliest}} \leq s_c \leq t_c^{\text{latest}}, \quad s_c + d_c - 1 \leq t_c^{\text{deadline}}, \quad (6)$$

where s_c is the selected start timestep. The term $s_c + d_c - 1$ is the last timestep occupied by a d_c -step cycle, so the final inequality ensures completion by the inclusive deadline. The start must also be compatible with the configured equipment and building limits. Once started, the cycle contributes

$$E_{c,t}^{\text{def}} = \begin{cases} q_{c,t-s_c}, & s_c \leq t < s_c + d_c, \\ 0, & \text{otherwise.} \end{cases} \quad (7)$$

Here $E_{c,t}^{\text{def}}$ is the cycle energy at timestep t and $q_{c,t-s_c}$ selects the corresponding element of the stored profile. Completed cycles, missed cycles, start delay, served energy and unserved energy are then exported as performance indicators.

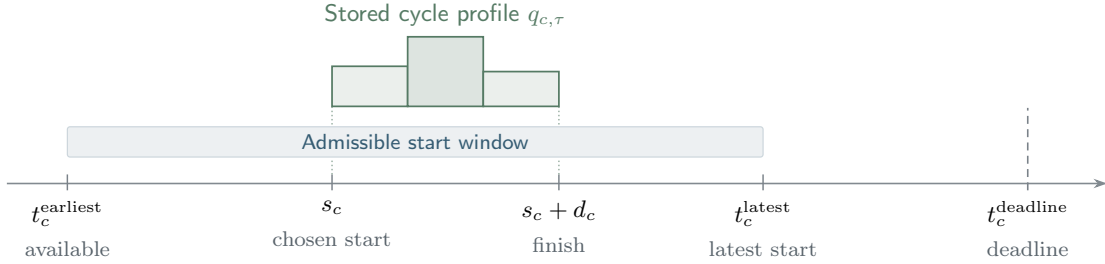


Figure 2.: Flexible-load start window, deadline and energy cycle.

Figure 2 relates the admissible start window to the stored cycle profile and completion deadline. The resulting service record identifies completed and missed cycles, start delays, and served or unserved energy.

3.6. Data and equipment failures

Requirement D6 concerns failures in the information and equipment used by a controller. CityLearn v3 distinguishes four types of data and equipment failures:

Measurement failures. These change the current values received from meters or sensors. A building-load measurement can, for example, become missing, noisy, biased, stuck at an earlier value or clipped to a specified range. The physical load remains unchanged; only the information received by the controller is affected.

Forecast failures. These change predicted values used by the controller, such as future PV generation, electricity prices or building demand. For example, a price forecast can be shifted by a fixed bias while the actual electricity price remains unchanged.

Action failures. These affect the command sent by the controller before it reaches the simulated equipment. A battery-charging command can, for example, be lost, delayed, biased,

noisy, stuck or clipped. The requested action remains recorded separately from the action that is actually applied.

Equipment failures. These make an item of equipment temporarily unavailable for measurement, control or both. For example, a battery or EV charger can remain physically present in the scenario while rejecting control actions during a declared failure interval.

CityLearn v3 stores these cases as time-stamped data and equipment failures, configured internally through the **robustness** block present in the schema. Each record states the affected channel, target, feature, start and end timesteps, failure mode and parameters.

The four failure types enter the control loop at different points. Let $\mathcal{E}_t^o$, $\mathcal{E}_t^f$, $\mathcal{E}_t^a$ and $\mathcal{E}_t^u$ be the active measurement, forecast, action and equipment-availability failures at timestep t . Their effect is

$$\begin{aligned}\tilde{o}_t &= \rho_o(o_t, \mathcal{E}_t^o, \mathcal{E}_t^u), & \tilde{f}_t &= \rho_f(f_t, \mathcal{E}_t^f), \\ a_t^{\text{app}} &= \rho_a(a_t^{\text{req}}, \mathcal{E}_t^a, \mathcal{E}_t^u), & x_{t+1} &= F(x_t, a_t^{\text{app}}, w_t).\end{aligned}\quad (8)$$

Here o_t and f_t are unchanged inputs, and $\tilde{o}_t$ and $\tilde{f}_t$ are the versions received by the controller. The maps ρ_o and ρ_f apply measurement and forecast failures. For compactness, ρ_a denotes command processing through failures, availability and feasibility checks, yielding a_t^{app} from a_t^{req} . The state-update function F advances physical state x_t using that applied action and exogenous inputs w_t , such as demand, weather and PV generation. A missing measurement therefore does not become a physical load reduction, and a lost command remains distinguishable from the requested action.

A short scenario can therefore describe common field problems without changing the controller implementation. The example used here declares a missing building-load measurement at timesteps 24–27, a biased price forecast at 48–55, lost storage commands at 72–75 and unavailable storage at 96–99. A fixed random seed makes stochastic changes repeatable, and a configured marker distinguishes a missing measurement from a physical zero. Figure 3 shows the same example as a timeline: measurement and forecast failures change what the controller receives, while action and equipment failures change what the simulator applies. The D6 row of Table A2 in Appendix A gives the exact fields of the internal **robustness** block and its event file.

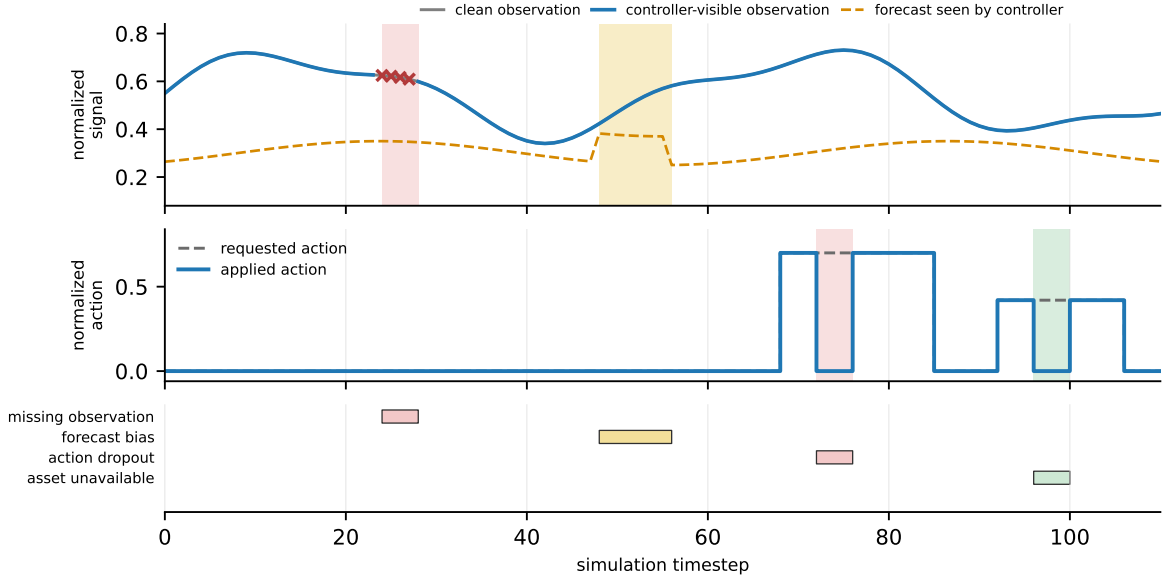


Figure 3.: Timeline of data and equipment failures in the controller loop.

The results report failure counts, active timesteps, changed measurements and forecasts, lost

commands and equipment-unavailable timesteps. These records link controller behaviour to the conditions under which it was tested.

3.7. Time resolution and energy units

Requirement D7 makes the simulation clock and energy units explicit. The timestep determines which variations the controller observes, when requests and deadlines occur, and how power is converted to energy. Hourly averages can hide short EV peaks, missing-data intervals or phase-limit exceedances. CityLearn v3 uses the configured timestep consistently across scenario timing, equipment accounting and evaluation; examples in this paper include hourly, 15 min, one-minute and 15 s data.

CityLearn v3 makes this clock explicit through the `seconds_per_time_step` property in the scenario schema and the declared simulation and episode bounds. Once loaded, request windows, service deadlines, actions, rewards and result exports use this timestep. If timestamps are available, the loader records their inferred spacing so that 15 s data are not silently interpreted as 15 min or 1 h data. The D7 row of Table A2 lists the exact clock and simulation-window fields.

The key conversion is simple. Let Δt be the simulation step duration in seconds. If a measured signal is provided as average power P_t in kW, the energy associated with one simulation step is

$$E_t = P_t \frac{\Delta t}{3600}, \quad (9)$$

where E_t is energy in kW h/step, P_t is average power in kW and $\Delta t/3600$ converts seconds to hours. Load, PV and charging time series expressed as energy are read directly in kW h/step; equipment ratings and import/export limits remain in kW; and prices and carbon intensities remain per kW h. Absolute measured series are distinguished from normalized profiles that must be scaled by capacity. For example, a 7.2 kW charger operating for one step represents 7.2 kW h at 1 h resolution, 1.8 kW h at 15 min and 0.03 kW h at 15 s.

3.8. Building and phase power limits

Requirement D8 concerns the power available at a building connection. A house, office or factory can have a contracted limit on the total power imported from or exported to the grid. A three-phase connection can also have a separate limit on each phase. Both types of limits must be respected at the same time. For example, a building can remain below its total import limit while exceeding the limit of the phase to which an EV charger is connected.

Phase assignment limits the power available to each device: a charger connected to L1 cannot use spare capacity on L2 or L3, whereas a balanced three-phase battery distributes its power across all three phases.

Consider a house with a total import limit of 12 kW and phase limits of 7 kW, 5 kW and 4 kW for L1, L2 and L3, respectively. Before any new control action, the house imports 4 kW on L1, 2 kW on L2 and 1 kW on L3, giving a total import of 7 kW. An EV charger connected to L1 then requests 7.2 kW. Accepting the complete request would increase total import to 14.2 kW and L1 import to 11.2 kW. The total connection has 5 kW of remaining capacity, but L1 has only 3 kW. The phase limit is therefore the restrictive one: the simulator applies 3 kW of charging and records that 4.2 kW of the request was rejected.

The same calculation is applied generally to all controllable electrical equipment. For building b , let Φ_b be its phase set: $\{L1\}$ for a single-phase connection or $\{L1, L2, L3\}$ for a three-phase connection. The set $\mathcal{U}_{b,t}$ contains the controllable equipment available at timestep t , such as EV chargers and stationary batteries. A request $u_{a,t}$ from equipment a is signed power in kW, with positive values increasing import and negative values increasing export. The fraction $m_{a,\phi}$ assigns this request to phase ϕ and sums to one across the phases used by the equipment. For

example, $m_{a,L1} = 1$ for a charger connected only to L1, while a balanced three-phase battery assigns one third of its power to each phase.

Before applying the limits, the requested total and phase powers are

$$P_{b,t}^{\text{req}} = P_{b,t}^0 + \sum_{a \in \mathcal{U}_{b,t}} u_{a,t}, \quad (10)$$

$$P_{b,\phi,t}^{\text{req}} = P_{b,\phi,t}^0 + \sum_{a \in \mathcal{U}_{b,t}} m_{a,\phi} u_{a,t}, \quad \phi \in \Phi_b. \quad (11)$$

Here $P_{b,t}^0$ is building power before new controllable requests, and $P_{b,\phi,t}^0$ is its phase allocation. These terms include non-controllable demand, local generation and flexible-load cycles that have already started.

Let I_b^{max} and X_b^{max} denote the total import and export limits. Their phase equivalents are $I_{b,\phi}^{\text{max}}$ and $X_{b,\phi}^{\text{max}}$. With positive power denoting import and negative power denoting export, an applied power state is feasible when

$$\begin{aligned} -X_b^{\text{max}} &\leq P_{b,t} \leq I_b^{\text{max}}, \\ -X_{b,\phi}^{\text{max}} &\leq P_{b,\phi,t} \leq I_{b,\phi}^{\text{max}}, \quad \forall \phi \in \Phi_b. \end{aligned} \quad (12)$$

An omitted limit is unbounded. Both total and phase constraints must hold; satisfying the total limit alone does not guarantee a feasible phase allocation.

When necessary, CityLearn v3 scales the controllable requests using factors $\alpha_{a,t} \in [0, 1]$. The resulting applied powers are

$$P_{b,t}^{\text{app}} = P_{b,t}^0 + \sum_{a \in \mathcal{U}_{b,t}} \alpha_{a,t} u_{a,t}, \quad (13)$$

$$P_{b,\phi,t}^{\text{app}} = P_{b,\phi,t}^0 + \sum_{a \in \mathcal{U}_{b,t}} \alpha_{a,t} m_{a,\phi} u_{a,t}. \quad (14)$$

A factor of one leaves a request unchanged, while a factor of zero rejects it completely. In the example, the charger receives $\alpha_{a,t} = 3/7.2 \approx 0.417$. Its applied power is therefore 3 kW, resulting in 10 kW of total building import and exactly 7 kW on L1. The original 7.2 kW request remains recorded alongside the applied value.

When non-controllable building power exceeds a limit, the simulator applies the available controllable reductions and records any remaining exceedance, retaining the original non-controllable load.

The results retain requested and applied actions, total and phase power, remaining capacity and rejected controllable power. Requested-action pressure is reported separately from residual limit violations calculated from applied power histories. Power exceedances are converted to energy using the timestep duration. This distinction separates an infeasible controller request from a physical exceedance that remains after controllable requests have been reduced.

Each building's optional `electrical_service` block declares single- or three-phase operation, total and per-phase import/export limits, default load allocation and equipment phase connections. Controllers may receive remaining total and phase capacity as inputs. When the block is absent, legacy CityLearn behaviour is retained unless older charger constraints are configured. Appendix A, Table A2, lists the fields.

4. Running and evaluating controller studies

Once the REC scenario is defined, a controller study must specify its inputs, reference policy, performance indicators and reporting procedure. This section addresses requirements S1–S5 in Table 3. Figure 4 summarizes the configuration choices from dataset and timestep to controller selection and result comparison.

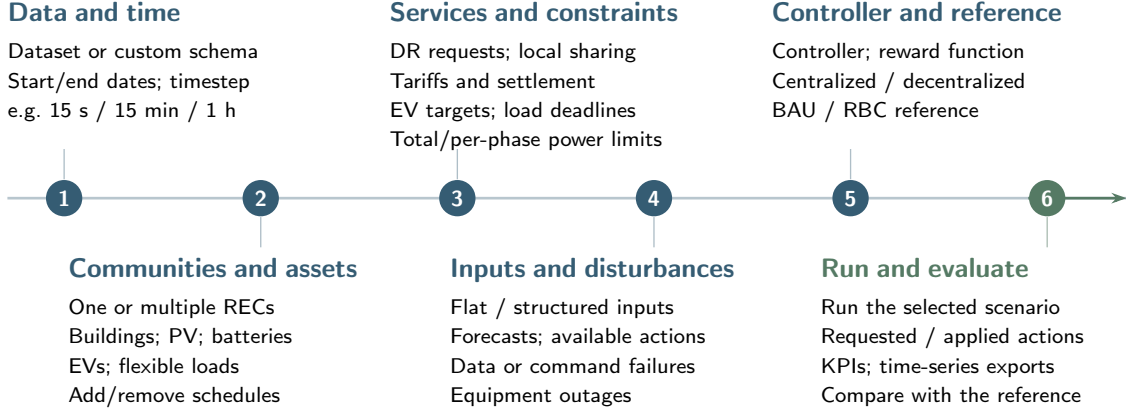


Figure 4.: Main choices in a CityLearn v3 experiment. The selected dataset provides initial settings; the user configures assets, services, constraints and control before running the study and comparing its results.

4.1. *Controller inputs and applied actions*

Requirement S1 concerns the meaning of controller inputs and actions throughout a run. When membership, equipment or service availability changes, a vector position alone may no longer identify the same entity. The interface must retain identity, availability and feasible actions.

CityLearn v3 keeps the original CityLearn loop: a controller resets the environment, receives inputs, submits actions through `step`, and receives a reward and run information. The scenario selects the input format and action-feedback settings together with the timestep, services, failures, reference controller, reward, performance indicators and exports.

Two input formats are supported. The flat format preserves the CityLearn/Gymnasium vector used by existing RBC, MPC and RL code in fixed configurations. The structured format, called *entity-based* in the implementation, retains the identifiers of buildings, equipment, EVs, chargers, flexible loads and requests, together with relations such as building–equipment and charger–EV. It also provides action ranges and availability masks. Dynamic member and asset changes require this structured interface. An installed charger appears as an identified row with its own action.

At each step, the simulator stores the requested action a_t^{req} and the action a_t^{app} applied after availability, service windows, electrical limits and command failures are considered. Availability masks and feasible ranges explain which actions were possible; the difference between requested and applied values records what the controller attempted and what the equipment executed.

4.2. *Performance indicators and result exports*

Requirement S2 is that performance indicators and result exports reveal service outcomes as well as aggregate performance. A controller can reduce cost while missing an EV departure, delaying a flexible load, failing a demand-response request or relying on action reduction by the simulator. Reward guides the controller, key performance indicators (KPIs) summarize the run, and exported trajectories show why those indicators changed.

CityLearn v3 retains the energy, cost, emissions, ramping and grid-impact indicators of CityLearn v2 and adds the families required by D1–D8. Table 4 uses the same names as Table 2; Appendix B provides the expanded catalogue.

Exports include KPI and time-series files, service summaries, requested/applied-action differences, entity tables, masks, relations and figure inputs. The companion CityLearn UI reads these folders directly; its trajectory and KPI views in Figure 5 support inspection of the records behind a scorecard.

Table 4.: Performance-indicator families in CityLearn v3.

Family	Example metrics	Why it matters
Energy and power limits	grid import/export, net exchange, peak, ramping, load factor, building/phase-limit exceedance	Retains district-control indicators and shows requests outside configured power limits.
Cost and settlement	energy cost, local settlement, comparison cost, savings, demand-response payments and penalties	Separates energy cost, local energy sharing and demand-response payments.
Emissions	emissions totals, daily averages and ratios to the reference controller	Keeps environmental performance visible alongside cost and service.
PV and local energy sharing	PV generation/export, self-consumption, local import/export, shared energy and sharing ratios	Shows local energy sharing instead of hiding it in aggregate net load.
EV service	departure counts, target/minimum/tolerance success, SOC deficit/surplus, charged and V2G energy	Distinguishes grid flexibility from mobility-service failure.
Flexible-load service	completed/missed cycles, service level, served/unserved energy and start delay	Evaluates flexible demand as a service with a deadline.
Demand-response requests	requests, active timesteps, requested/credited/shortfall energy, compliance, payments and invalid baselines	Measures requested flexibility and its settlement.
Data and equipment failures	failures, changed measurements/forecasts/actions, missing values, lost commands and unavailable equipment	Records the conditions under which controller results were produced.
Changing community members and assets	active periods, time-stamped changes and changing controller-input/action definitions	Links member and asset changes to valid actions and result periods.
Controller inputs and actions	table rows, relations, masks, feasible capacity and requested/applied-action differences	Shows what the controller knew, requested and physically caused.
Multiple communities	local rows, summed quantities and weighted ratios	Keeps each community visible next to combined results.
Inherited thermal/HVAC/comfort	discomfort ratios, hot/cold deltas, thermal-resilience and unserved-energy indicators	Retains the thermal, HVAC and comfort indicators of CityLearn v2.

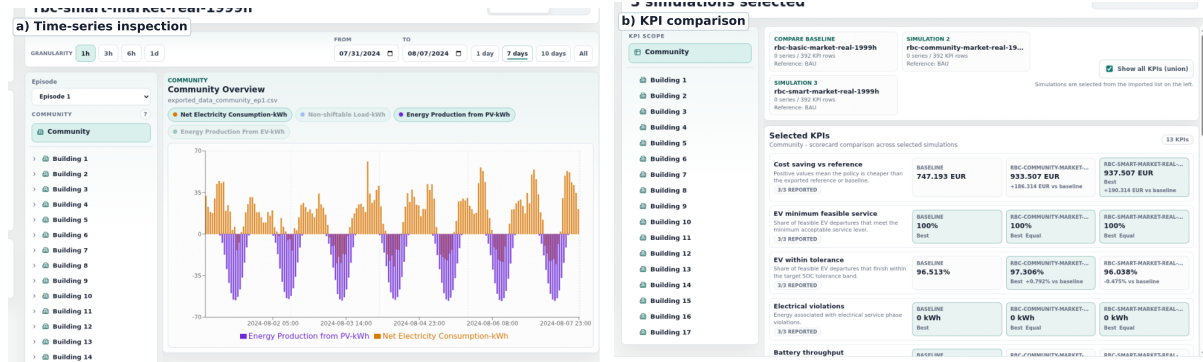


Figure 5.: Time-series and KPI comparison views in the CityLearn UI.

The scorecard should match the study: EV service and action reductions for mobility, or traded energy, settlement and participant costs for local sharing. Compact scorecards support comparison, while the exports retain the full indicator families for further analysis.

4.3. Reference controllers and fair comparisons

Requirement S3 is a declared reference case evaluated under the same scenario and accounting rules. Matching service obligations, settlement and electrical limits provides a consistent basis for normalized comparisons between policies.

The reference controller must therefore state how it charges EVs, starts mandatory flexible loads, operates batteries and reacts to demand-response requests or failures. It uses the same settlement and building/phase-limit rules as the evaluated controller.

Table 5.: Reference controllers used in the CityLearn v3 evaluation workflow.

Policy class	What it represents	Interpretation in REC studies
<code>NormalNoBatteryPolicy</code>	Day-to-day operation for EVs and flexible loads without stationary battery control.	Separates native service demand from the value of stationary storage.
<code>NormalPolicy</code>	BAU operation with the same native service logic plus simple PV self-consumption storage.	Serves as the main reference line for annual scorecards and normalized KPI interpretation.
<code>RBCBasicPolicy</code>	Service-aware rule-based control with simple price response.	Checks whether a transparent controller can improve cost or import while tracking EV targets more precisely than normal charging.
<code>RBCSmartPolicy</code>	Solar-, price- and peak-aware rule-based control.	Provides a stronger internal RBC reference for grid, carbon and service KPIs.
<code>RBCCommunityPolicy</code>	REC-aware rule-based control using community surplus and import context.	Tests whether local sharing and settlement objectives can be inspected separately from grid and carbon outcomes.

KPIs can then be reported in raw form and, when appropriate, relative to the declared reference controller b . A generic ratio and difference are

$$\kappa_j^{\text{ratio}}(\pi, b) = \frac{\kappa_j(\pi)}{\kappa_j(b)}, \quad (15)$$

$$\Delta\kappa_j(\pi, b) = \kappa_j(\pi) - \kappa_j(b), \quad (16)$$

where π is the evaluated controller, b is the reference controller and κ_j is indicator j . The ratio is defined only when $\kappa_j(b) \neq 0$. Service failures, invalid demand-response baselines, shortfalls and power-limit exceedances remain visible as raw counts, rates or gates instead of being hidden by normalization.

4.4. Software validation

Requirement S4 concerns checks on scenario rules, interfaces and reported results. CityLearn v3 combines regression tests with targeted REC tests, time-unit and physical audits, controller-input checks, BAU/KPI consistency checks and thermal/HVAC compatibility tests. Table 6 reports their scope and observed outcomes. The checks were run in June 2026 using Python 3.10.12. The test logs and machine-readable audit outputs are retained with the manuscript.

Table 6.: Software validation checks used in this manuscript revision.

Check	Protected failure mode	Current evidence
Full regression suite	General simulator behaviour and compatibility with existing CityLearn tests.	423 passed tests; 18 warnings.
Targeted REC requirements	KPI exports, demand-response requests, data and equipment failures, structured controller inputs, changing community members and assets, and runs with multiple communities.	103 passed tests; 8 warnings.
Physics and time-unit audit	Energy/power consistency when the same scenario is evaluated at 60, 300, 900 and 3600 s timesteps.	16/16 scenarios passed.
Controller-input and change audit	Persistent member/equipment identities, relations, action rows and configuration versions when equipment or members change.	Strict pass.
BAU/KPI sanity audit	Reference-controller rows, duplicated KPI keys, finite KPI values and delta-to-BAU identities in controller exports.	3317 rows per export, 1401 BAU-derived rows and no duplicate KPI keys.
Thermal/HVAC compatibility tests	Inherited thermal and heat-pump behaviour alongside the new REC capabilities.	12 passed tests.

Each figure in Section 5 is linked to its scenario descriptor, command, raw output folder, KPI table and figure input. These records identify the timestep, controller inputs, reference controller, events and KPI scope behind each plot.

4.5. Computational cost

Requirement S5 concerns the cost of repeated controller experiments. High-frequency data increase storage and loading requirements, structured inputs enlarge the returned data, and synchronized communities increase the state and indicators processed at each step.

The comparisons in Table 7 were run on an Intel Core i9-13980HX processor (24 cores, 32 threads), with 29.1 GiB of RAM available to Linux, using Python 3.10.12 and Linux 6.8.0-124 on x86_64. Control actions were zero and rendering was disabled. Step durations were measured around `env.step()` using `perf_counter`, excluding controller decision time. The basic flat/structured comparison reports means over five process-isolated runs per configuration (seeds 0–4), each with 599 executed hourly steps. The detailed-input and multi-community cases use one 599-step hourly run per configuration; the CSV/Parquet cases use one 239-step run per configuration at 15 s resolution. Initialization and reset were timed separately.

Table 7.: Computational-cost evidence.

Check	Compared cases	Observed change	Interpretation
Basic structured input	Matched building/phase-limit run, flat input versus the basic structured input.	5.30 to 5.51 ms/step; about +0.21 ms/step or +4.1%.	Persistent identities, relations, masks and action feedback are available with measured but small basic overhead.
High-frequency storage	Two 15 s CSV/Parquet pairs: power limits and changing community members and assets.	2.446 GB to 12.405 MB and 2.619 GB to 12.740 MB; initialization falls from 25.37 to 15.61 s and from 28.59 to 18.92 s.	Parquet makes both datasets about 200 times smaller and reduces initialization time by 34–38%; post-load step time remains similar.
Structured-input size	Flat input versus the most detailed standard structured input.	4.88 to 10.18 ms/step; 165 flat values to 2538 table values and 302 feature names.	Step time approximately doubles while the exposed input payload becomes more than 15 times larger.
Multiple communities	One, two and four synchronized communities in the heterogeneous run.	5.02, 7.08 and 11.41 ms/step; four communities export 5227 KPI rows, including 107 combined rows.	Four synchronized communities require 2.27 times the single-community step time while retaining local and combined outcomes.

The basic structured interface adds 4.1% to mean step time. The most detailed input exposes over 15 times as many values at 2.09 times the step time. Four communities average 11.41 ms/step, approximately 88 steps/s, with 5227 local and combined KPI rows.

5. Application examples

The application examples examine how scenario conditions affect service delivery, visible peaks, settlement and controller records. Table 8 summarizes the questions and experimental configurations.

The scenario configurations are summarized in Appendix A.

Table 8.: Application-example protocol used in this manuscript.

Example	Study question	Scenario/controller	Reported outcome
Reference controllers	Can BAU and transparent reference controllers be interpreted with the expanded KPI families on a full annual REC/EV scenario?	8760-step hourly REC scenario with EVs, flexible loads, local energy sharing and building/phase power limits; legacy EV-RBC, BAU and rule-based controllers.	The scorecard compares energy, cost, EV and flexible-load service, settlement and requested power-limit exceedances.
Time resolution and energy units	Can the same energy trace imply different visible power peaks at different time resolutions?	One synthetic annual 15 s REC trace viewed at 1 h, 15 min and 15 s; all views are derived from the same physical signal.	Equal annual energy can produce different visible peaks, so the declared timestep matters for capacity-sensitive studies.
Local energy sharing and settlement	Does local settlement change the annual economic interpretation of a community-aware controller?	Full-year RBCCommunityPolicy ; grid-only comparison versus configured local settlement for the same trajectory.	Local exchange changes final cost and member outcomes and cannot be inferred from district net load alone.
Demand-response requests	Can a time-window request be translated into delivery and settlement evidence?	180-step scenario with three non-overlapping requests; one evening request shown against operation without a request.	Request-level results expose requested energy, credited response, shortfall and payment.
Changing community members and assets	Do time-stamped changes propagate to controller inputs, actions and results?	One run in which a member, chargers and PV equipment are added or removed; detailed structured-input results are archived.	Member/equipment counts and controller rows change together while identities remain traceable.
Data and equipment failures	Do repeatable failures change the controller trajectory and appear in the results?	Same controller and period in a clean run and a run with declared measurement, forecast, command and equipment failures.	Declared failures produce identifiable changes in controller inputs, applied actions and exported results.
Multiple communities	Can heterogeneous communities be evaluated together without hiding individual outcomes?	Four heterogeneous communities evaluated over the same time window.	Individual profiles and combined outcomes quantify simultaneous surplus and demand across communities.

5.1. Reference controllers and scorecard interpretation

The first example combines energy, cost, service, settlement and electrical-constraint indicators in an annual scorecard. It compares the reference policies defined in Section 4.3 across grid, economic and service outcomes.

The hourly three-phase REC/EV scenario contains EV chargers, one flexible-load service, local sharing and building/phase limits. Its annual window has 8760 timesteps and 8759 executed control steps. Six policies are compared: the older **Legacy EV-RBC** and the five reference policies in Table 5. **NormalPolicy** supplies the BAU reference, combining ordinary EV and flexible-load service with simple PV self-consumption storage.

Figure 6 gives each indicator as a ratio to BAU, with arrows indicating the comparison direction. EV target proximity measures the proportion of eligible departures within ± 5 percentage points of the requested SOC, while minimum service measures attainment of the configured minimum acceptable SOC. The six policies share 2757 eligible departures, the same settlement rules and the same building/phase limits. BAU attains the minimum in all eligible departures but lies within the symmetric target band in 6.3%, because its charging rule aims for 100% SOC and frequently exceeds the requested target.

Annual scorecard: controller / BAU (dimensionless)

Green / red: along / against $\uparrow \downarrow$; saturation at $\pm 40\%$

Cost $\downarrow$	1.47	1.05	BAU	0.85	0.78	0.77
Grid / REC import $\downarrow$	1.38	1.05	BAU	0.87	0.81	0.80
Peak $\downarrow$	1.06	1.08	BAU	1.16	1.00	1.00
Emissions $\downarrow$	1.28	1.05	BAU	0.86	0.77	0.76
EV target proximity (within ± 5 pp) $\uparrow$	1.75	1.00	BAU	15.4x	14.4x	14.4x
EV minimum service $\uparrow$	0.80	1.00	BAU	1.00	0.94	0.94
Flexible-load service $\uparrow$	1.00	1.00	BAU	1.00	1.00	1.00
REC export $\downarrow$	2.43	1.24	BAU	1.29	1.11	1.05
Settlement savings $\uparrow$	1.06	1.03	BAU	0.92	0.82	0.81
Requested limit exceedance $\downarrow$	9.8kx	0.00	BAU	0.09	0.00	0.00
	Legacy EV-RBC	Normal no battery	BAU (v3)	RBCBasic Policy	RBCSmart Policy	RBCCommunity Policy

Figure 6.: Annual REC/EV controller scorecard normalized to the CityLearn v3 BAU reference.

Legacy EV-RBC produces an annual cost of 42.2 kEUR, grid import of 230.1 MWh and emissions of 38.4 tCO₂, compared with 28.7 kEUR, 166.8 MWh and 30.0 tCO₂ for BAU. It meets the minimum feasible EV target for 80.1% of feasible departures and accumulates 49.5 MWh of requested power-limit exceedance energy.

RBCBasicPolicy reduces annual cost to 24.4 kEUR and grid import to 145.4 MWh while retaining 100% minimum EV service and reaching 98.0% target proximity. Its peak rises from 104.7 kW for BAU to 121.4 kW. RBCSmartPolicy reduces cost to 22.3 kEUR, import to 135.2 MWh and emissions to 23.1 tCO₂. RBCCommunityPolicy reaches 22.0 kEUR, 133.3 MWh, a 104.6 kW peak and 22.8 tCO₂. The two smart policies attain target proximity of 91.4% and 91.2%, respectively, with minimum EV service of 93.8% for both and flexible-load service of 100%. Among the RBC policies, Basic delivers full minimum EV service, while Smart and Community achieve the lowest cost and import. The target-proximity ratios compare the frequency of departures within the requested SOC band. Settlement savings reflect the timing and volume of local exchange.

Figure 7 follows one charger through a 24 h window. BAU begins charging when the vehicle connects, while the RBC policies shift part of the request to later hours with different PV output and prices. The middle panel shows connection status, requested departure SOC and the SOC at timestep boundaries, including the state reached after the final control action. Smart and Community finish at 91.0% SOC against a 92% target, within the configured tolerance. The lower panel supplies the building's PV and price context.

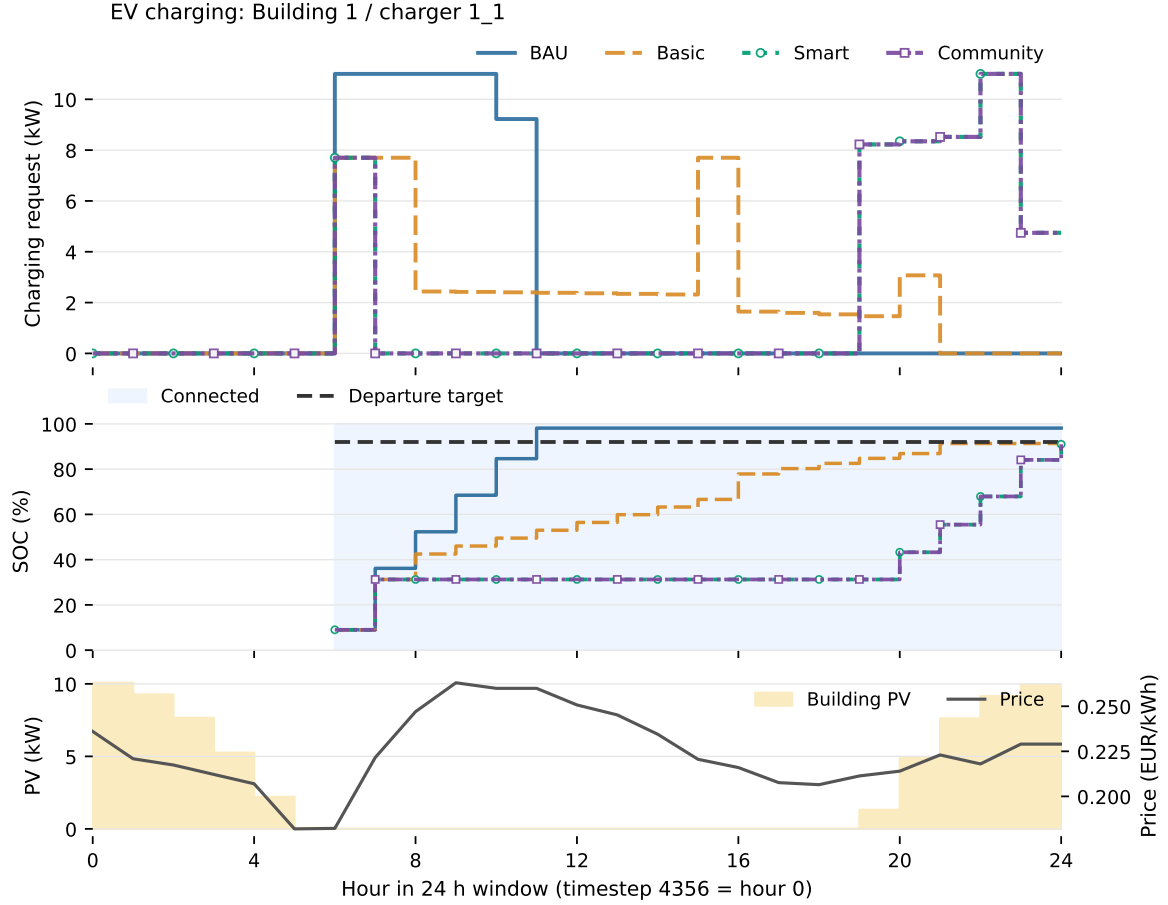


Figure 7.: EV charging requests over 24 h and the corresponding timestep-boundary SOC states for one charger.

5.2. Time resolution and hidden peak visibility

Short timesteps retain brief EV-charging and process-load peaks that hourly averaging can conceal. This experiment uses a deterministic replay of one synthetic annual 15 s REC trace and constructs 15 min and 1 h views by averaging the same signal. All views retain 349.0 MWh of load, 147.6 MWh of PV generation and 19.3 MWh of EV charging, isolating the effect of temporal aggregation on visible power.

Figure 8 shows the consequence on a selected day and over the annual trace. On the selected day, the native 15 s trace reaches 231.0 kW when a short high-power event occurs. The same day reaches only 104.8 kW in the 15 min averaged view and 63.9 kW in the hourly view, so the hourly representation hides most of the short event. Over the annual trace, the hourly view reports an observed peak of 96.8 kW and a 99.9th percentile power of 78.2 kW. The 15 min view reports 188.2 kW and 107.7 kW, while the native 15 s view reports 281.5 kW and 136.1 kW. Thus, the same annual energy balance can imply substantially different visible peaks and upper-tail behaviour depending on the timestep used by the controller and evaluator.

Aggregation conceals short capacity-relevant events even when annual energy is unchanged. The declared timestep therefore affects the peaks and upper-tail power available to a controller or evaluator.

High-frequency data reveal short peaks hidden by coarser averages

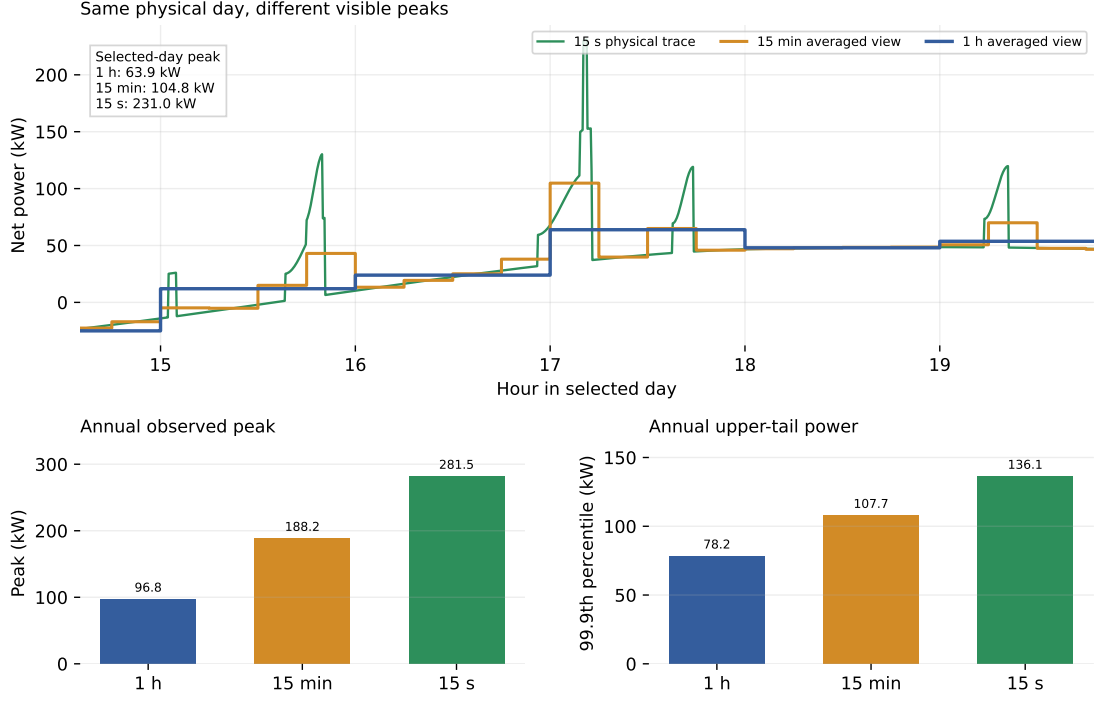


Figure 8.: High-frequency peak visibility from one synthetic 15 s trace.

5.3. Local energy sharing and settlement

This experiment values the annual `RBCCCommunityPolicy` trajectory from Section 5.1 under grid-only accounting and configured REC settlement. Both valuations use the same import, export, PV, battery and service trajectories over 8760 hourly timesteps, with 8759 executed control steps.

Under grid-only accounting, the annual electricity cost of the `RBCCCommunityPolicy` trajectory is 24.8 kEUR. With local settlement, the same trajectory is valued at 22.0 kEUR, a reduction of 2.8 kEUR or 11.3%. The run records 17.0 MWh of locally traded energy, 133.3 MWh of residual grid import and 48.3 MWh of residual grid export. Local exchange covers 11.3% of demand and 26.1% of export energy under the configured rule. All 17 members reduce annual cost, with a median saving of 121 EUR and a range from 68.8 to 525.7 EUR. These savings accompany 91.2% EV target proximity, 93.8% minimum EV service and 100% flexible-load service for the same controller.

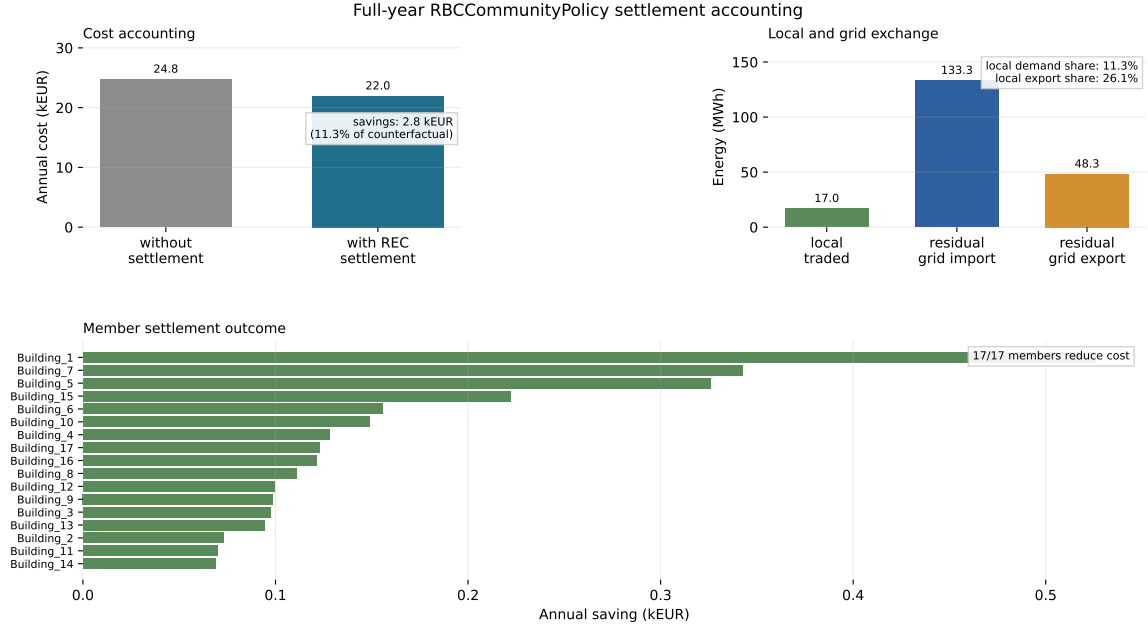


Figure 9.: Full-year REC settlement accounting for the same RBCCCommunityPolicy trajectory.

The range of member savings shows how local settlement distributes community-level benefits. The participant-level exports provide the quantities needed to compare allocation rules and fairness, including the authors' related simulator-based analysis (Fonseca et al. 2026).

5.4. Demand-response delivery and settlement

This experiment applies an event-aware dispatch to explicit demand-response requests and tracks delivery, credit, shortfall and settlement.

The configured 180-step scenario contains three non-overlapping requests, active over 10 hourly timesteps and requesting 190.0 kWh in total. Figure 10 focuses on one evening down request. In the upper panel, the shaded region marks the active window. The red step curve asks for 25 kWh of reduction per timestep, or 100.0 kWh in total. The blue markers show the response credited for payment. It follows the request closely for the first three timesteps and falls below it in the final timestep, creating a shortfall.

The lower panel compares settlement with a no-request counterfactual, whose DR-related cost change remains zero. The green curve falls as credited response is remunerated; negative values indicate lower net cost relative to that counterfactual. A shortfall penalty reduces the benefit near the end of the event. The selected request finishes with 90.6 kWh of credited response, 7.4 kWh of shortfall and 25.2 EUR lower net cost.

Across the three requests in Table 9, event-aware dispatch reduces total shortfall from 101.1 to 51.6 kWh, raises compliance from 0.43 to 0.70 and lowers cost after settlement by 73.0 EUR relative to reference operation. Net revenue remains negative after shortfall penalties. Delivered energy records all movement in the requested direction, while compliance uses credit capped at each timestep. Credit and penalized shortfall use different thresholds because tolerance reduces the latter. These records connect the change in net cost to the response delivered during each request.

Table 9.: Combined demand-response results for the three-request scenario.

Case	Requested kWh	Delivered kWh	Shortfall kWh	Compliance ratio	Net DR revenue EUR	Cost after settlement EUR
Reference operation	190.0	82.5	101.1	0.43	-100.2	863.6
Event-aware dispatch	190.0	300.9	51.6	0.70	-16.0	790.6

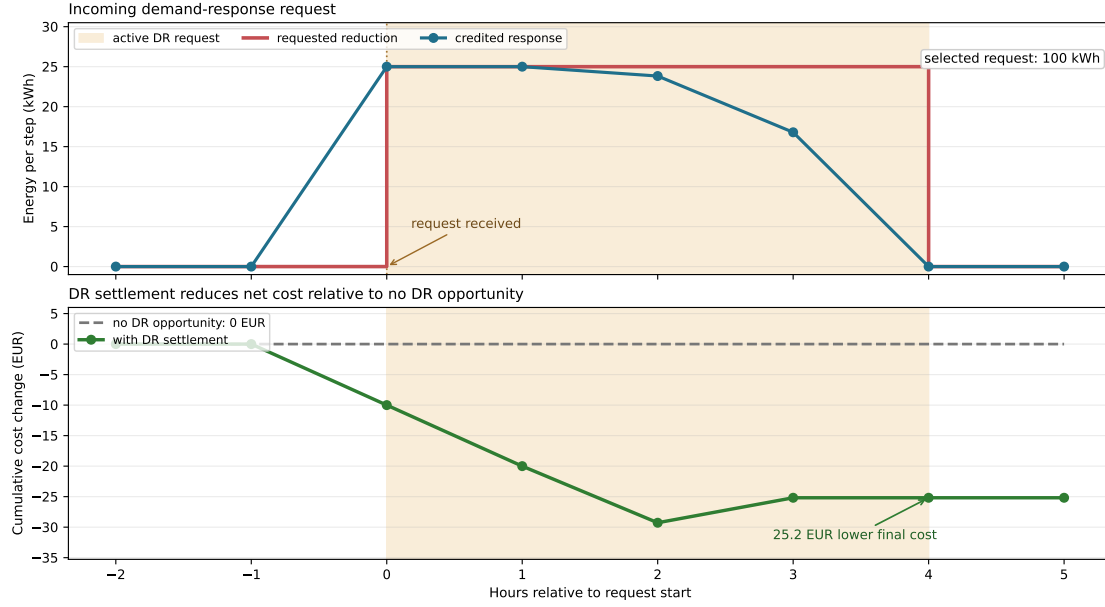


Figure 10.: Selected demand-response request and settlement opportunity.

5.5. Changing community members and assets

This example tests whether member and equipment changes propagate to the structured controller input within one continuous run.

Four changes occur in one run: Building 18 is added at timestep 1000, a charger is added to Building 2 at 1300, a charger is removed from Building 5 at 1500 and PV is removed from Building 11 at 1600. Figure 11 shows the resulting counts. Active members increase from 17 to 18, active chargers increase from 8 to 10 and then return to 9, and active flexible loads increase from 1 to 2 when the new member enters.

The structured input changes at the same timesteps: identified rows increase from 67 to 74 and end at 71, relations move from 74 to 84 and then 80, and action rows move from 26 to 30 and then 29. The largest input contains 2295 table values. Persistent identifiers allow these changes to be traced to individual members and assets. Archived input sets range from 85 feature names in the basic set to 302 in the largest standard set.

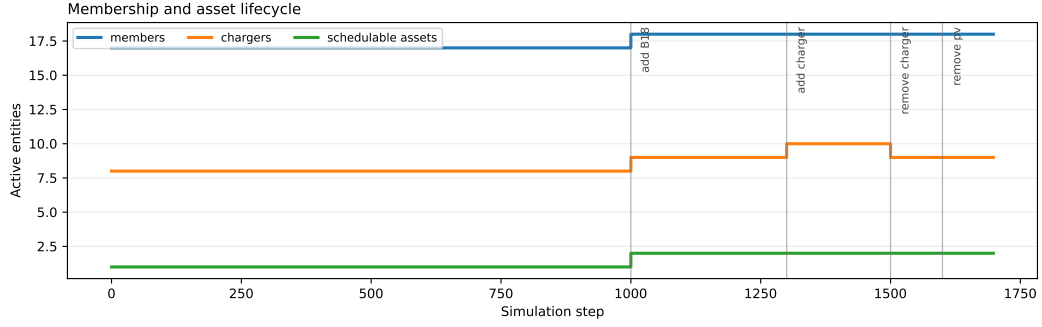


Figure 11.: Counts of changing community members and assets.

5.6. Data and equipment failures

The same hourly week is run twice with a storage policy: once with unchanged inputs and equipment availability, and once with declared failures. The policy produces non-zero storage requests around the failure windows so that lost commands and unavailable equipment leave visible traces in the trajectory and service records.

The file declares four failures over 20 active timesteps: missing building-load measurements, biased electricity-price forecasts, lost storage commands and temporary storage unavailability. Figure 12 places them on the same simulation axis and reports the associated counts. The missing-meter interval creates 4 records, the forecast-bias interval creates 8, the command failure affects 136 action entries across the storage actions, and the equipment failure records 68 unavailable equipment–timestep pairs. The figure shows what was declared, when it occurred and how it was counted.

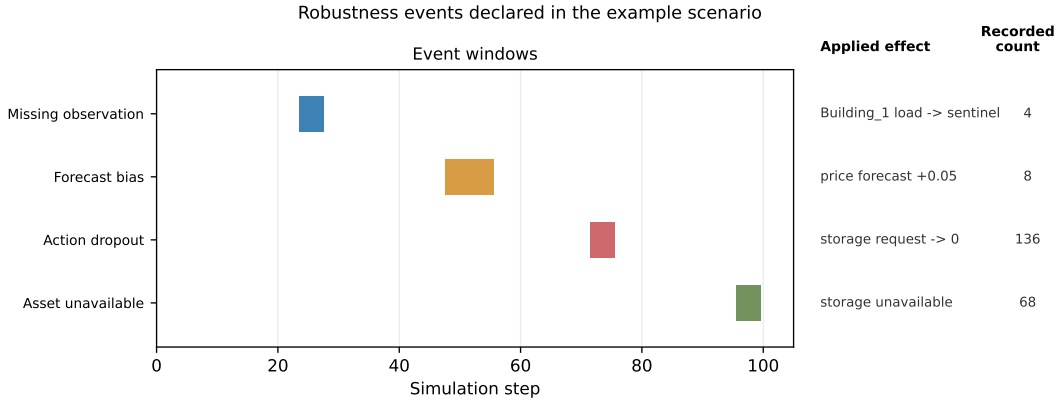


Figure 12.: Data and equipment failures in the example scenario.

Figure 13 shows the effects of measurement and forecast failures. On the left, the clean trace shows the physical meter value. During the failure, the controller-visible value becomes the configured missing-value marker, -9999 , rather than a physical zero. The crosses at the bottom mark the missing-measurement interval. On the right, affected forecast values are shifted by $+0.05$ EUR/kWh. The controller thus receives corrupted information while the export records when and how it was changed.

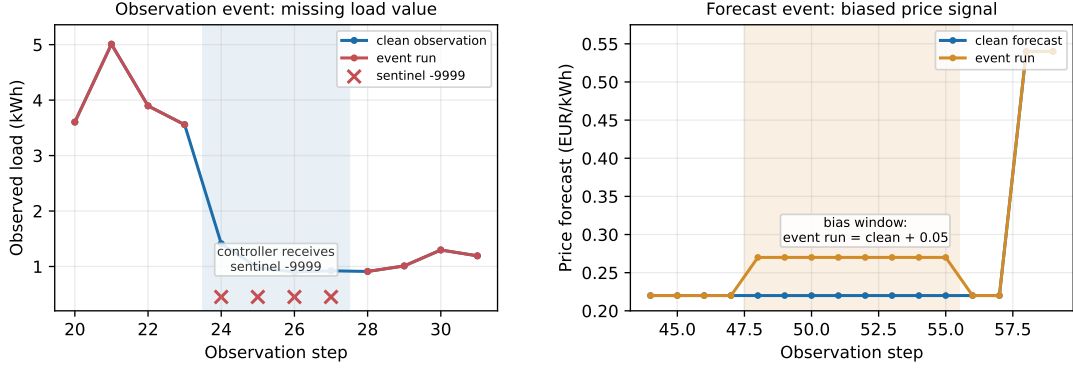


Figure 13.: Missing measurements and biased forecasts.

The same run also records lost commands and unavailable equipment. During the command-failure window, requested storage actions remain recorded while the corresponding applied actions become zero; mean net electricity is then 25.33 kWh/step higher than in the clean run. Over the full run, the grid-import and cost ratios increase by approximately 0.011 and 0.0041, while the peak ratio is unchanged. The event records link these changes to the affected measurement, forecast, command and equipment channels.

5.7. Multiple communities

Synchronized experiments can reveal complementary demand and generation profiles while retaining each community's objectives and results.

The experiment evaluates four REC scenarios over the same 600-step hourly window. REC 1 has many EVs; RECs 2–4 have different building demand and PV profiles. Each electrically independent community is controlled with *RBCSmartPolicy*. After the run, post-processing matches simultaneous export in one community with import in another under a common accounting rule. The matched quantities measure inter-community sharing opportunities from the recorded trajectories.

Figure 14 shows the result in two complementary ways. The left side plots a high-sharing window. REC 1 repeatedly exports while the other RECs remain importers, yet the aggregate profile of all four RECs is still positive. The separate REC profiles reveal simultaneous local surplus and demand beneath a positive aggregate net load.

The right side gives the full-run accounting view. Over the 600-step window, 2.45 MWh of local export is matched with demand in other communities, and no residual export remains after this matching calculation. Of that matched demand, 0.79 MWh of REC 2 import, 0.76 MWh of REC 3 import and 0.87 MWh of REC 4 import are matched mainly with REC 1 export; small additional matches grow from the other RECs to REC 3.

Under the normalized accounting rule, locally matched energy is valued at 80% of the retail import price and grid export remuneration is zero. This gives recipient-side savings of 0.16, 0.16 and 0.17 normalized cost units for the three receiving communities, and 0.49 normalized cost units in total. The combined records quantify the potential savings associated with simultaneous surplus and demand across the four communities.

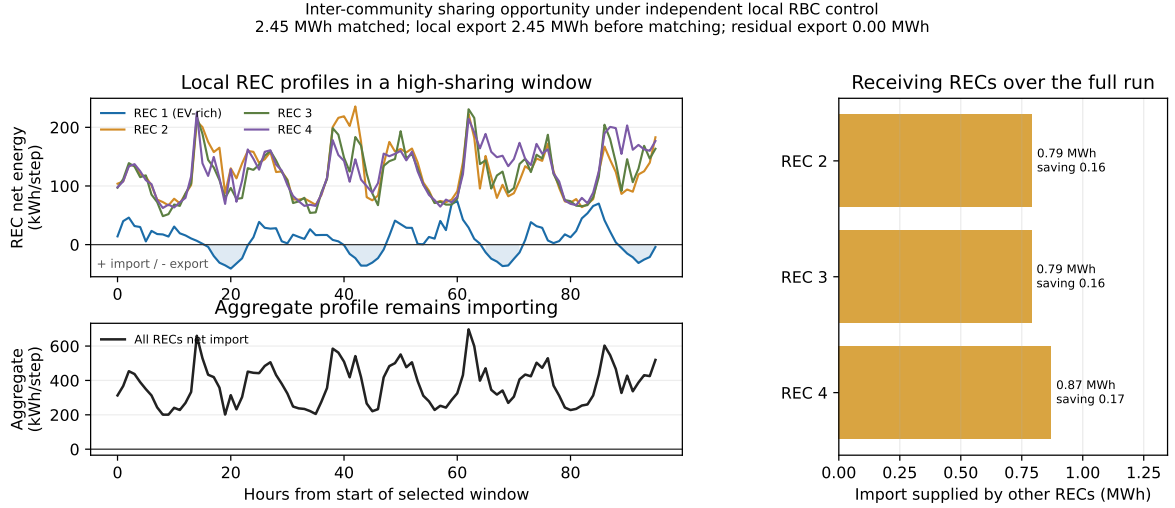


Figure 14.: Inter-community sharing opportunity under independent local control.

6. Discussion and conclusions

This paper presented CityLearn v3 as a configurable framework for studying control in renewable energy communities whose participants, equipment and operating conditions can change over time. Scenario configuration determines both what the controller observes and how its decisions are applied, with service requirements and electrical limits carried through to evaluation. Building and phase active-power limits are enforced alongside timestep energy balances, while local sharing and settlement connect physical operation to participant costs. This extends the building-energy and thermal modelling capabilities of CityLearn v2 into a common environment for realistic REC control experiments.

The application results demonstrate why energy, service and economic indicators need to be read together. In the annual controller comparison, lower electricity costs and grid imports coincide with different levels of EV departure service. Local settlement, in turn, changes participant costs without changing the physical trajectory, and demand-response remuneration depends on both delivered response and shortfall. Temporal resolution also shapes the interpretation of a trajectory: averaging the same 15 s trace to 15 min or 1 h preserves energy totals but conceals short power peaks.

For controller research, the value of this framework lies in being able to vary those conditions while retaining a consistent link between decisions and outcomes. Requested and applied actions explain how electrical limits or equipment availability affect operation, and service records identify the consequences for community members. The open-source implementation and public example scenarios provide a basis for extending these studies to new policies and configurations. CityLearn v3 thus supports comparisons that connect energy and economic performance to the decisions, constraints and delivered services that produce it.

Acknowledgements

This work was supported by Fundação para a Ciência e a Tecnologia, I.P. (FCT), under the supervision and superintendence of the Ministry of Education, Science and Innovation, through doctoral grant 2024.00855.BD, financed by the Portuguese State Budget and co-financed by the European Social Fund Plus (FSE+) through the Programa Demografia, Qualificações e Inclusão (PDQI/PESSOAS 2030). This work was also supported by the DEMFLEX operation (COMPETE2030-FEDER-01657200 – 19215), supported by the Innovation and Digital Transition Programme (COMPETE 2030), under Portugal 2030, and co-financed by the European

Union. The views and opinions expressed in this paper are those of the author alone and do not necessarily reflect the views of FCT, the European Union, KDT JU, or the respective funding and granting authorities. Neither these institutions nor the granting authorities can be held responsible for them.

Disclosure statement

No potential conflict of interest was reported by the authors.

Data and code availability statement

CityLearn v3 source code and public example scenarios are available from the official repository, <https://github.com/citylearn-project/CityLearn>.

References

- Belmar, Francisco, Patrícia Baptista, and Diana Neves. 2023. "Modelling renewable energy communities: assessing the impact of different configurations, technologies and types of participants." *Energy, Sustainability and Society* 13 (18). <https://doi.org/10.1186/s13705-023-00397-1>.
- Blum, David, Javier Arroyo, Sen Huang, Ján Dragoňa, Filip Jorissen, Harald Taxt Walnum, Yan Chen, et al. 2021. "Building optimization testing framework (BOPTTEST) for simulation-based benchmarking of control strategies in buildings." *Journal of Building Performance Simulation* 14 (5): 586–610. <https://doi.org/10.1080/19401493.2021.1986574>.
- Crawley, Drury B., Linda K. Lawrie, Frederick C. Winkelmann, W. F. Buhl, Y. J. Huang, Curtis O. Pederesen, Richard K. Strand, et al. 2001. "EnergyPlus: creating a new-generation building energy simulation program." *Energy and Buildings* 33 (4): 319–331. [https://doi.org/10.1016/S0378-7788\(00\)00114-6](https://doi.org/10.1016/S0378-7788(00)00114-6).
- Eurogia2030. 2026. "DEMFLEX: Demonstration of AI-Enhanced Energy Flexibility for Microgrids with Connected Smart Homes." Running projects catalogue. Accessed 16 June 2026, <https://eurogia.eu/running-projects/>.
- European Commission, CORDIS. 2025. "OPEVA - OPTimization of Electric Vehicle Autonomy." CORDIS project record 101097267. Last update 4 August 2026; accessed 9 September 2026, <https://cordis.europa.eu/project/id/101097267>.
- European Parliament and Council of the European Union. 2018. "Directive (EU) 2018/2001 of the European Parliament and of the Council of 11 December 2018 on the promotion of the use of energy from renewable sources." Official Journal of the European Union, L 328, 82–209. Consolidated version accessed 18 June 2026, <https://eur-lex.europa.eu/eli/dir/2018/2001/oj/eng>.
- European Parliament and Council of the European Union. 2019. "Directive (EU) 2019/944 of the European Parliament and of the Council of 5 June 2019 on common rules for the internal market for electricity." Official Journal of the European Union, L 158, 125–199. Consolidated version accessed 18 June 2026, <https://eur-lex.europa.eu/eli/dir/2019/944/oj/eng>.
- Fonseca, T., C. Sousa, L. Ferreira, P. Rodrigues, P. Paiva, R. Venâncio, R. Severino, and L. Matos. 2026. "Can intelligent Renewable Energy Communities deliver on equity for a just energy transition? A policy oriented demonstrator analysis." *Energy Research & Social Science* 133: 104584. <https://doi.org/10.1016/j.erss.2026.104584>.
- Fonseca, Tiago, Luis Lino Ferreira, Bernardo Cabral, Ricardo Severino, Kingsley Nweye, Dipanjan Ghose, and Zoltan Nagy. 2025. "EVLearn: extending the CityLearn framework with electric vehicle simulation." *Energy Informatics* 8 (16). <https://doi.org/10.1186/s42162-024-00445-w>.
- Frieß, Nathalie, Elias Feiner, Ulrich Pfersch, Joachim Schauer, and Thomas Strametz. 2023. "Optimization and Simulation for the Daily Operation of Renewable Energy Communities." Preprint, Optimization Online. Accessed 16 June 2026, https://optimization-online.org/wp-content/uploads/2023/09/preprint_EC_s.pdf.
- Gjorgievski, Vladimir Z., Snezana Cundeva, and George E. Georghiou. 2021. "Social arrangements, technical designs and impacts of energy communities: A review." *Renewable Energy* 169: 1138–1156. <https://doi.org/10.1016/j.renene.2021.01.078>.

- Hardy, Trevor D., Bryan Palmintier, Philip L. Top, Dheepak Krishnamurthy, and Jason C. Fuller. 2024. “HELICS: A Co-Simulation Framework for Scalable Multi-Domain Modeling and Analysis.” *IEEE Access* 12: 24325–24347. <https://doi.org/10.1109/ACCESS.2024.3363615>.
- Hernandez-Matheus, Alejandro, Markus Löschenbrand, Kjersti Berg, Ida Fuchs, Mònica Aragüés-Peñalba, Eduard Bullich-Massagué, and Andreas Sumper. 2022. “A systematic review of machine learning techniques related to local energy communities.” *Renewable and Sustainable Energy Reviews* 170: 112651. <https://doi.org/10.1016/j.rser.2022.112651>.
- Mello, João, Luís Rodrigues, José Villar, and João Saraiva. 2024. “Energy allocation and settlement in collective self-consumption.” In *2024 20th International Conference on the European Energy Market (EEM)*, 1–6. IEEE.
- Mousavi Motlagh, Farzaneh, Thien-An Nguyen-Huu, Pieter-Jan Hoes, Trung Thai Tran, Jan Hensen, and Phuong Hong Nguyen. 2023. “Simulation-based design optimization of local energy communities: a case study of the BAM living lab.” In *Proceedings of Building Simulation 2023: 18th Conference of IBPSA*, .
- Nweye, Kingsley, Kathryn Kaspar, Giacomo Buscemi, Tiago Fonseca, Giuseppe Pinto, Dipanjan Ghose, Satvik Duddukuru, et al. 2025. “CityLearn v2: Energy-flexible, resilient, occupant-centric, and carbon-aware management of grid-interactive communities.” *Journal of Building Performance Simulation* 18 (1): 17–38. <https://doi.org/10.1080/19401493.2024.2418813>.
- OPEVA Consortium. 2026. “OPEVA: OPTimization of Electric Vehicle Autonomy.” Project website. Accessed 16 June 2026, <https://opeva.eu/>.
- SoftCPS Laboratory. 2025. “EnerGAize: AI-Powered Energy Management for Renewable Energy Communities.” Project news page. Accessed 16 June 2026, <https://www2.isep.ipp.pt/softcps/?p=1112>.
- SoftCPS Laboratory. 2026. “DEMFLEX: Demonstration of AI-Enhanced Energy Flexibility for Microgrids with Connected Smart Homes.” Project page. COMPETE20-30-FEDER-01657200-19215; accessed 16 June 2026, <https://www2.isep.ipp.pt/softcps/?p=1235>.
- Steinbrink, Cornelius, Marita Blank-Babazadeh, Andre El-Ama, Stefanie Holly, Bengt Lueers, Marvin Nebel-Wenner, Rebeca P. Ramirez Acosta, et al. 2019. “CPES Testing with mosaik: Co-Simulation Planning, Execution and Analysis.” *Applied Sciences* 9 (5): 923. <https://doi.org/10.3390/app9050923>.
- Towers, Mark, Ariel Kwiatkowski, Jordan Terry, John U. Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, et al. 2024. “Gymnasium: A Standard Interface for Reinforcement Learning Environments.” arXiv preprint arXiv:2407.17032. <https://arxiv.org/abs/2407.17032>.
- Vázquez-Canteli, José R., Sourav Dey, Gregor Henze, and Zoltan Nagy. 2020. “CityLearn: Standardizing research in multi-agent reinforcement learning for demand response and urban energy management.” *arXiv preprint arXiv:2012.10504* <https://doi.org/10.48550/arXiv.2012.10504>, <https://arxiv.org/abs/2012.10504>.
- Vázquez-Canteli, José R., and Zoltan Nagy. 2019. “Reinforcement learning for demand response: A review of algorithms and modeling techniques.” *Applied Energy* 235: 1072–1089. <https://doi.org/10.1016/j.apenergy.2018.11.002>.
- Wetter, Michael, Wangda Zuo, Thierry S. Nouidui, and Xiufeng Pang. 2014. “Modelica Buildings library.” *Journal of Building Performance Simulation* 7 (4): 253–270. <https://doi.org/10.1080/19401493.2013.765506>.

Appendix A records the validation and application configurations; Appendix B expands the KPI definitions; and Appendix C maps implementation areas to the capabilities discussed in the paper.

Appendix A. Scenario evidence and configuration reference

A.1. Scenarios used as evidence

Table A1.: Software-validation configurations and application-evidence scenarios.

Scenario	Temporal scope	Equipment represented	Changing conditions	Paper role
Base REC	3600 s; 48-step KPI check, 600-step runtime check and full-year reference-controller scorecard	Buildings, PV, batteries, EV chargers and flexible loads	none	BAU/KPI checks and the annual controller scorecard.
EV departure target	3600 s; scorecard service result	EVs, chargers, PV, batteries and buildings	none	Checks strict target, tolerance and minimum-feasible EV departure indicators.
Flexible-load schedules and deadlines	3600 s; scorecard service result	Buildings and flexible appliances	Start windows and deadlines	Checks completed/missed cycles and flexible-load service indicators.
Demand-response requests	3600 s; 48-step KPI check and 180-step application experiment	Buildings, PV, batteries and EV chargers	Demand-response request file	Checks request, delivery, shortfall and settlement indicators.
Data and equipment failures	3600 s; 48-step KPI check and hourly-week application experiment	Buildings, PV, batteries and EV chargers	Data and equipment failure file (robustness internally)	Checks measurement, forecast, command and equipment-availability failures.
Changing community members and assets	60, 300, 900 and 3600 s; 160-step audits, plus an hourly application with changes at steps 1000–1600	Buildings, chargers, PV, batteries and flexible loads	Member and equipment changes	Checks identities, active periods, controller inputs and result windows.
Multiple communities	3600 s; 48-step KPI check and 600-step combined run	Four heterogeneous REC scenarios	none in current check	Checks individual and combined KPI rows and matched inter-community energy.
15 s Parquet	15 s; file-size, 240-step storage/runtime check and 24 h BAU comparison	Power-limit and changing-equipment demonstrations	optional	Checks storage footprint, loading and time-resolution-aware BAU accounting.
Synthetic annual trace	Native 15 s annual replay and 15 min/1 h averages	Aggregate load, PV and EV traces	None	Quantifies the effect of temporal aggregation on visible power.

A.2. Exact configuration reference for Section 3

Table A2 lists the scenario keys and linked-file fields for requirements D1–D8 in the order used in Table 2. Internal names and legacy aliases are included where relevant.

Table A2.: Exact configuration reference for the eight REC requirements in Section 3.

ID and requirement	Scenario declaration	Linked file or record fields and supported values
D1 Multiple communities	<code>MultiCommunityEnv</code> receives <code>communities</code> ; each entry contains <code>community_id</code> , <code>schema</code> , optional <code>env_kwargs</code> and optional <code>weight</code> . Wrapper output uses optional <code>render_directory</code> and <code>render_session_name</code> .	<code>community_id</code> is unique; <code>weight</code> is finite and non-negative, with at least one positive weight. Child scenarios must use matching <code>seconds_per_time_step</code> , episode length, interface and central-agent settings.
D2 Demand-response requests	Top-level <code>demand_response</code> : <code>enabled</code> , <code>requests_file</code> , <code>baseline_method</code> , <code>baseline_window_seconds</code> and <code>allow_overlapping_requests</code> . The current baseline mode is <code>rolling_pre_event_average</code> ; overlapping requests are not currently supported.	Request-file columns: <code>request_id</code> , <code>issuer</code> , <code>direction</code> , <code>start_time_step</code> , <code>end_time_step</code> , <code>target_power_kw</code> , <code>activation_price_eur_per_kwh</code> , <code>shortfall_penalty_eur_per_kwh</code> and optional <code>tolerance_power_kw</code> . Issuer is <code>dso</code> or <code>tso</code> ; direction is <code>up</code> or <code>down</code> .

ID and requirement	Scenario declaration	Linked file or record fields and supported values
D3 Local energy sharing and settlement	Top-level <code>community_market: enabled</code> , <code>local_price_ratio_to_grid_import</code> , optional <code>import_member_weights</code> , <code>kpis.community_local_traded_enabled</code> and <code>kpis.community_self_consumption_enabled</code> .	The price ratio is clipped to $[0,1]$; <code>intra_community_sell_ratio</code> is accepted as a legacy alias. Residual grid export is recorded but has zero remuneration in the current implementation; it has no active configurable settlement price.
D4 Changing community members and assets	Top-level <code>topology_mode</code> and <code>topology_events</code> . Dynamic changes require <code>topology_mode=dynamic</code> and the entity controller interface.	Event fields: <code>id</code> , <code>time_step</code> , <code>operation</code> , <code>target_member_id</code> , <code>target_asset_type</code> , <code>target_asset_id</code> , optional <code>source_member_id</code> , <code>source_asset_id</code> and overrides. Operations are <code>add_member</code> , <code>remove_member</code> , <code>add_asset</code> and <code>remove_asset</code> ; asset types are <code>charger</code> , <code>deferrable_appliance</code> , <code>p_v</code> and <code>electrical_storage</code> .
D5 Flexible-load schedules and dead-lines	Under <code>buildings.<id>.deferrable_appliances.<id></code> : <code>cycle_profiles_file</code> , <code>flexibility_schedule_file</code> and optional <code>attributes.trigger_threshold</code> .	Cycle-profile columns: <code>profile_id</code> , <code>duration_steps</code> , <code>total_energy_kwh</code> , <code>load_profile</code> . Schedule columns: <code>cycle_id</code> , <code>profile_id</code> , <code>earliest_start_time_step</code> , <code>latest_start_time_step</code> , <code>deadline_time_step</code> , <code>priority</code> , <code>must_run</code> . The <code>must_run</code> field is the mandatory-service flag described in the main text.
D6 Data and equipment failures	Top-level data and equipment failure block (<code>robustness</code> internally): <code>enabled</code> , <code>events_file</code> , <code>random_seed</code> , <code>missing_replacement_value</code> and <code>modules.<observations forecasts actions assets>.enabled</code> .	Event-file columns: <code>event_id</code> , <code>module</code> , <code>target_type</code> , <code>target_id</code> , <code>target_feature</code> , <code>start_time_step</code> , <code>end_time_step</code> , <code>mode</code> , with optional <code>value</code> , <code>std</code> , <code>min_value</code> , <code>max_value</code> , <code>replacement_value</code> and <code>delay_steps</code> . Modes depend on the channel: <code>missing/noise/bias/stuck/clip</code> for measurements and forecasts; <code>dropout/noise/bias/stuck/delay/clip</code> for commands; unavailable for equipment.
D7 Time resolution and energy units	Top-level <code>seconds_per_time_step</code> , <code>simulation_start_time_step</code> , <code>simulation_end_time_step</code> and <code>episode_time_steps</code> . Episode selection can additionally use <code>rolling_episode_split</code> and <code>random_episode_split</code> .	<code>seconds_per_time_step</code> is a positive duration such as 3600, 900, 60 or 15 s. Time-series columns declared as power remain in kW; energy columns are in kW h/step. Linked request, deadline and failure fields use integer timestep indices on the same clock.
D8 Building and phase power limits	Under <code>buildings.<id>.electrical_service: mode</code> , <code>default_split</code> , <code>limits.total.import_kw</code> , <code>limits.total.export_kw</code> , <code>limits.per_phase.<L1 L2 L3>.import_kw</code> and <code>export_kw</code> ; optional <code>observations</code> switches expose headroom, violations and phase encoding. Equipment uses <code>attributes.phase_connection</code> .	<code>mode</code> is <code>single_phase</code> or <code>three_phase</code> ; <code>default_split</code> is <code>balanced</code> , <code>L1</code> , <code>L2</code> or <code>L3</code> where compatible; <code>phase_connection</code> is <code>L1</code> , <code>L2</code> , <code>L3</code> or <code>all_phases</code> . Omitted power limits are unbounded.

A.3. Related service and evaluation configuration

The following settings define EV service, controller inputs, applied actions and result exports.

Table A3.: Related service and evaluation configuration.

Item	Main fields or files	Purpose
EV service	EV connection and departure schedules, charger power limits, V2G availability, target/minimum state of charge and departure-service fields	Preserves mobility requirements next to grid objectives so a low cost or peak does not hide a missed departure target.
Controller inputs and applied actions	<code>interface</code> , structured-input sets, action masks and action feedback	Defines what the controller receives and keeps requested and applied actions separate.
Performance indicators and result exports	KPI/export settings, output fields and CSV or Parquet result format	Defines the indicators, trajectories and diagnostics retained for comparison and audit.

Appendix B. KPI catalogue and equations

This appendix summarizes the KPI families in Table 4, with equations where they clarify scope or normalization.

Table B1.: Expanded KPI catalogue for CityLearn v3.

Family	KPI	Definition / rationale	Unit
Energy and power limits	Net electricity consumption	$E_{b,t}^{\text{net}}$ or $E_{c,t}^{\text{net}}$ after equipment operation and local accounting.	kWh/step
Energy and power limits	Net-boundary grid import	$\sum_t \max(E_t^{\text{grid}}, 0)$ for signed energy at the stated boundary; distinguish this from summed member imports.	kWh
Energy and power limits	Net-boundary grid export	$\sum_t \max(-E_t^{\text{grid}}, 0)$ at the same boundary; distinguish this from summed member exports.	kWh
Energy and power limits	Net grid energy	Grid import minus grid export.	kWh
Energy and power limits	Average daily peak	Mean of daily maximum net import power.	kW
Energy and power limits	Peak import	Maximum net grid import power over the evaluation window.	kW
Energy and power limits	Peak export	Maximum net grid export power over the evaluation window.	kW
Energy and power limits	Ramping	Sum or mean absolute changes in net load over the evaluation window.	kW or kWh
Energy and power limits	Load factor	Average import divided by peak import over the evaluation window.	ratio
Energy and power limits	One minus load factor	Complement of load factor when lower-is-better normalization is used.	ratio
Energy and power limits	Zero-net-energy ratio	Evaluated-controller net energy relative to the declared reference policy; the exported metric specifies the aggregation scope.	ratio
Energy and power limits	Requested-pressure / residual-violation counts	Separate counts of requests beyond configured electrical limits and violations remaining in the applied power state.	count
Cost and settlement	Electricity cost	Energy import/export/local settlement multiplied by configured tariffs.	currency
Cost and settlement	Local export credit	Locally shared export multiplied by the local price; residual grid export is unremunerated in the described implementation.	currency
Cost and settlement	Net energy cost	Import cost minus export/local revenues, using the configured settlement.	currency
Emissions	Carbon emissions	$\sum_t E_t^{\text{grid-import}} c_t$, where c_t is carbon intensity.	kgCO ₂
Emissions	Import carbon intensity	Emissions divided by imported energy.	kgCO ₂ /kWh
Inherited thermal/HVAC/comfort	Heat-pump electricity	Electricity consumed by heating/cooling heat-pump devices in scenarios with those devices.	kWh
	Unmet thermal demand	Heating/cooling demand not served, if represented in the selected scenario.	kWh or ratio
	Comfort violation hours	Timesteps or hours outside configured thermal comfort bounds.	time
	Occupant override count	Number of thermostat/comfort overrides, if occupant feedback is active.	count
	HVAC clipping count	Number of heat-pump/HVAC actions clipped by configured limits.	count
PV and local energy sharing	PV generation	Total measured or simulated PV energy.	kWh
PV and local energy sharing	PV self-consumption	Locally consumed PV divided by PV generation.	ratio
PV and local energy sharing	Self-sufficiency	Demand served by local resources divided by total demand.	ratio
PV and local energy sharing	Local import	Energy imported by members from local community surplus.	kWh
PV and local energy sharing	Local export	Energy supplied by members for local sharing.	kWh
PV and local energy sharing	Local exchange	Matched local surplus and demand before residual grid exchange.	kWh

Family	KPI	Definition / rationale	Unit
PV and local energy sharing	Local sharing ratio	Local exchange divided by eligible surplus or demand according to the exported demand-side or export-side variant.	ratio
PV and local energy sharing	Community surplus	Energy remaining after local matching and demand satisfaction.	kWh
PV and local energy sharing	Community deficit	Energy not served by local resources before grid import.	kWh
PV and local energy sharing	Settlement cost/revenue	Member or community settlement under local and grid prices.	currency
EV service	EV connected time-step count	Number of timesteps with EV connected and controllable/observable.	count
EV service	EV arrival count	Number of EV arrivals in the evaluation window.	count
EV service	EV departure count	Number of EV departures in the evaluation window.	count
EV service	Charged energy	Energy delivered to EV batteries.	kWh
EV service	Discharged energy / V2G energy	Energy discharged from EVs to building/community/grid.	kWh
EV service	Required departure energy	Energy required to reach target SOC at departure.	kWh
EV service	Departure success count	Number of departures meeting target SOC within tolerance.	count
EV service	Departure success ratio	Successful departures divided by total evaluated departures.	ratio
EV service	Departure deficit energy	$\sum_i \max(E_i^{\text{target}} - E_{i,d_i}^{\text{stored}}, 0)$, where i identifies a departure, d_i is its timestep and the two energy terms are required and stored EV energy.	kWh
EV service	Maximum individual departure deficit	Maximum deficit observed for a departing EV.	kWh
EV service	Mean departure deficit	Mean deficit across departures.	kWh
EV service	Average required charging power	Required energy divided by remaining connection time.	kW
EV service	Slack time / urgency	Time remaining after accounting for required charging at feasible power.	time
EV service	Charger clipping count	Number of EV actions clipped by charger/battery/electrical constraints.	count
Flexible-load service	Requested cycles	Number of flexible-load cycles requested by the scenario.	count
Flexible-load service	Started cycles	Number of requested cycles started.	count
Flexible-load service	Completed cycles	Number of requested cycles completed before deadline.	count
Flexible-load service	Missed cycles	Number of cycles not completed by deadline.	count
Flexible-load service	Service level	Completed cycles divided by requested cycles.	ratio
Flexible-load service	Cycle energy served	Energy consumed by completed or active cycles.	kWh
Flexible-load service	Unserved cycle energy	Requested cycle energy not delivered by deadline.	kWh
Flexible-load service	Average start delay	Mean delay relative to earliest feasible start.	time
Flexible-load service	Maximum start delay	Maximum delay among served cycles.	time
Flexible-load service	Deadline violation count	Number of cycle deadlines missed or violated.	count
Demand-response requests	Request count	Number of demand-response requests in the evaluation window.	count
Demand-response requests	Active timestep count	Number of timesteps inside active request windows.	count
Demand-response requests	Requested total energy	$\sum_{(r,t) \in \mathcal{W}} R_r \Delta t / 3600$, where $\mathcal{W}$ contains valid active request-timestep pairs and Δt is the timestep in seconds.	kWh
Demand-response requests	Delivered total energy	$\sum_{(r,t) \in \mathcal{W}} d_{r,t} \Delta t / 3600$, retaining delivery opposite to the requested direction as a negative value.	kWh

Family	KPI	Definition / rationale	Unit
Demand-response requests	Credited total energy	$\sum_{(r,t) \in \mathcal{W}} c_{r,t} \Delta t / 3600$, where $c_{r,t}$ is the non-negative capped credit in Equation (2).	kWh
Demand-response requests	Shortfall total energy	$\sum_{(r,t) \in \mathcal{W}} s_{r,t} \Delta t / 3600$, with $s_{r,t}$ defined in Equation (2).	kWh
Demand-response requests	Compliance ratio	Credited energy divided by requested energy over valid request timesteps.	ratio
Demand-response requests	Revenue total	Credited energy multiplied by the activation price.	currency
Demand-response requests	Penalty total	Shortfall energy multiplied by the penalty price.	currency
Demand-response requests	Net revenue total	Revenue minus penalties.	currency
Demand-response requests	Invalid baseline timestep count	Number of request timesteps without a valid baseline.	count
Demand-response requests	Up/down delivered energy	Signed delivered energy separated by request direction.	kWh
Data and equipment failures	Failure count	Number of configured failures activated.	count
Data and equipment failures	Changed-measurement count	Number of measurement entries affected by missing, noise, bias, stuck or clipping modes.	count
Data and equipment failures	Changed-forecast count	Number of forecast entries affected.	count
Data and equipment failures	Missing-measurement count	Number of measurement values set to the missing marker.	count
Data and equipment failures	Changed-action count	Number of commands modified by dropout, noise, bias, stuck, delay or clipping modes.	count
Data and equipment failures	Lost-command count	Number of commands dropped or replaced.	count
Data and equipment failures	Equipment-unavailable count	Number of equipment–timestep pairs marked unavailable.	count
Data and equipment failures	Failure summary	Count or weighted count by affected channel and failure type.	mixed
Changing community members and assets	Active member count	Number of active REC members per timestep or average over the evaluation window.	count
Changing community members and assets	Active asset count	Number of active assets per timestep or average over the evaluation window.	count
Changing community members and assets	Change count	Number of member or asset additions/removals.	count
Changing community members and assets	Active-period energy	Energy and KPIs computed only over active periods.	mixed
Controller inputs and actions	Input table count	Number of structured input tables returned by the interface.	count
Controller inputs and actions	Input row count	Number of identified rows across input tables.	count

Family	KPI	Definition / rationale	Unit
Controller inputs and actions	Relation count	Number of exposed relations, e.g., building-equipment or charger-EV.	count
Controller inputs and actions	Action mask count	Number of unavailable/infeasible action entries.	count
Controller inputs and actions	Reduced-action count	Number of requested actions reduced by physical or service limits.	count
Controller inputs and actions	Feasible action capacity	Available feasible charge/discharge or start capacity exposed to the controller.	mixed
Multiple communities	Combined grid import/export	Sum of community import/export.	kWh
Multiple communities	Combined cost and emissions	Sum of community cost/emissions.	mixed
Multiple communities	Weighted combined ratio	Weighted mean of ratios such as self-consumption or service level.	ratio
Multiple communities	Community-level KPI table	Indicators retained for each community before combination.	mixed
Multiple communities	Combined request/failure/change totals	Sum of demand-response requests, failures and member/equipment changes across communities.	count

Appendix C. Implementation map

Table C1.: Implementation areas supporting the scenario and evaluation capabilities.

Area	Scope	Role in the paper
Mobility and flexible services	EVs, chargers and appliance cycles	Represents charging obligations and flexible service windows.
Core simulation	Buildings, storage, PV, control loop, established KPIs and datasets	Provides the simulation basis for the REC requirements in Section 2.
Software distribution	Official CityLearn repository and versioned releases	Distributes the simulator and its public examples.
Control loop and physical corrections	Terminal states, action parsing, SOC, storage and high-frequency timestep handling	Supports explicit scenarios and comparable controller studies.
Performance indicators and exports	Runtime, loading, export and KPI separation	Supports auditable KPI calculation and result inspection.
Structured inputs and changing members/assets	Identified controller inputs and changing active periods	Supports structured control and changing community members and assets.
Time resolution and Parquet	Power/energy units, measured PV and high-frequency data loading	Supports measured data and large datasets.

Area	Scope	Role in the paper
Flexible-load schedules and deadlines	Cycle profiles and allowed start windows	Supports flexible loads beyond washing machines.
Demand-response requests	File-based requests and settlement indicators	Supports demand-response evaluation.
Multiple communities	Synchronized runs and weighted combination	Supports several independent RECs with individual and combined KPIs.
Data and equipment failures	Changes to measurements, forecasts, commands and availability	Supports repeatable failure testing.