Unit Commitment

Contents

The **unit_commitment** extension adds commitment-level operational constraints to selected technologies: online/started/stopped unit accounting, minimum output floors, maximum output ceilings (replacing the default capacity constraint), and minimum up-time and down-time windows. It also supports discounted startup costs and startup emissions and input flows.

It is disabled by default and enabled per run through configuration:

```
extensions = ["unit_commitment"]
```

Conceptual Model

The core idea is that the continuous capacity variable $\mathbf{CAP}_{r,p,t,v}$ is *notionally divided* into discrete units, each of size $UC_{r,t}$. The number of units available is therefore $\mathbf{CAP}_{r,p,t,v}/UC_{r,t}$. Crucially, $\mathbf{CAP}$ itself remains a standard linear (continuous) Pyomo variable — the extension does not change its domain. In practice, because the UC constraints tie output directly to the integer count of online units, the optimiser will typically land on a solution where $\mathbf{CAP}$ is an integer multiple of $UC_{r,t}$, but this is a consequence of the optimisation rather than an explicit constraint.

The three UC variables — $\mathbf{UCN}$ (online), $\mathbf{UCST}$ (started), and $\mathbf{UCSP}$ (stopped) — count whole units and are promoted to non-negative integers by default (MIP). In **linearized mode** (`linearized = 1`), these variables are kept continuous. This is equivalent to imagining that $UC_{r,t}$ is infinitesimally small: the commitment variables become fractional occupancy factors rather than discrete unit counts, and the min-output/max-output bounds act as simple lower and upper capacity-factor constraints without any integer requirements. Linearized mode is useful for reducing solve time during early model development or sensitivity analysis.

> ⚠ Warning
>
> **Annual technologies are incompatible with unit commitment.** Technologies in the `tech_annual` set have no time-slice flexibility and cannot be committed; the model will raise an error if any annual technology appears in the `unit_commitment` table.

⚠ Warning

Integer vs. linear relaxation. By default all three UC variables ($\mathbf{UCN}$, $\mathbf{UCST}$, $\mathbf{UCSP}$) are promoted to non-negative integers, turning the problem into a MIP. Set `linearized = 1` in the `unit_commitment` table for a particular technology to keep those variables continuous (LP relaxation).

ℹ Note

The `uc_max_output_constraint` **replaces** the default `capacity_constraint` for processes listed in the `unit_commitment` table. Non-UC processes retain the standard capacity constraint unchanged.

Overview

Table	Purpose	Key interaction
`unit_commitment`	Core UC parameters (unit size, output fractions, up/down times).	Enables UC constraints; replaces capacity constraint for UC techs. Commitment results are written to `output_unit_commitment`.
`unit_commitment_startup_cost`	Direct startup cost per unit of capacity started.	Added to variable costs.
`unit_commitment_startup_emissions`	Emissions produced at startup per unit of capacity started.	Added to `limit_emission_constraint` totals, emission costs and `output_emission` table.
`unit_commitment_startup_input`	Input commodity consumed at startup per unit of capacity started.	Added to consumption in `commodity_balance_constraint` and results in `output_flow_in`.

Parameters

unit_commitment

$UC_{r \in R,\, t \in T}$

The primary UC table. Each row registers a technology (or technology group) as subject to unit-commitment constraints and specifies all operational parameters. Exactly one row per (r, t) pair is required.

Column	Symbol	Default	Description
`region`	r	—	region label
`tech`	t	—	technology name
`unit_capacity`	$UC_{r,t}$	—	nameplate capacity per discrete unit (same units as `v_capacity`); **required**
`min_output_fraction`	$\underline{f}_{r,t}$	0	minimum output as a fraction of available nameplate output per online unit; $[0, 1]$
`max_output_fraction`	$\overline{f}_{r,t}$	1	maximum output as a fraction of available nameplate output per online unit; $(0, 1]$
`min_up_time_hours`	$T^{up}_{r,t}$	0	minimum number of consecutive hours a unit must remain online after starting
`min_down_time_hours`	$T^{dn}_{r,t}$	0	minimum number of consecutive hours a unit must remain offline after stopping
`linearized`	—	0	if 1, UC variables are kept continuous (LP relaxation); if 0, they are promoted to non-negative integers (MIP)

unit_commitment_startup_cost

$UCSC_{r \in R,\, t \in T}$

Optional. Specifies a cost incurred each time a unit is started, expressed per unit of capacity. The total startup cost in a timeslice (s, d) is:

$$\text{startup cost} = \mathbf{UCST}_{r,p,s,d,t,v} \cdot UC_{r,t} \cdot UCSC_{r,t}$$

Column	Symbol	Description
`region`	r	region label
`tech`	t	technology name
`cost_per_cap`	$UCSC_{r,t}$	startup cost per unit of capacity (same currency units as `cost_invest`)

unit_commitment_startup_emissions

$UCSE_{r \in R,\, e \in C^e,\, t \in T}$

Optional. Specifies an emission produced at startup per unit of capacity started. Startup emissions are summed into the `limit_emission_constraint` alongside normal activity-based emissions.

Column	Symbol	Description
`region`	r	region label
`emis_comm`	e	emission commodity
`tech`	t	technology name
`emis_per_cap`	$UCSE_{r,e,t}$	emission per unit of capacity started (same units as `emission_activity`)

unit_commitment_startup_input

$UCSI_{r \in R,\, i \in C^p,\, t \in T}$

Optional. Specifies a physical commodity consumed at startup per unit of capacity started. Startup inputs are summed into the `commodity_balance_constraint` alongside normal flow-based consumption.

Column	Symbol	Description
`region`	r	region label
`input_comm`	i	input commodity
`tech`	t	technology name
`input_per_cap`	$UCSI_{r,i,t}$	input commodity consumed per unit of capacity started

Decision Variables

Three UC decision variables are added per (r, p, s, d, t, v) index. By default they are non-negative integers (MIP); set `linearized = 1` to relax them to continuous non-negative reals.

Variable	Domain	Description
$\mathbf{UCN}_{r,p,s,d,t,v}$ (`v_uc_online`)	$\mathbb{Z}_{\geq 0}$	number of units online at the start of timeslice (s, d)
$\mathbf{UCST}_{r,p,s,d,t,v}$ (`v_uc_started`)	$\mathbb{Z}_{\geq 0}$	number of units that start up during timeslice (s, d)
$\mathbf{UCSP}_{r,p,s,d,t,v}$ (`v_uc_stopped`)	$\mathbb{Z}_{\geq 0}$	number of units that shut down during timeslice (s, d)

Constraints

Unit Count Bounds

temoa.extensions.unit_commitment.components.commitment.**uc_online_upper_constraint**(*model: UnitCommitmentModel, r: Region, p: Period, s: Season, d: TimeOfDay, t: Technology, v: Vintage*) → ExprLike [source]

Online units cannot exceed the number of available (physical) units:

$$\mathbf{UCN}_{r,p,s,d,t,v} \leq \frac{\mathbf{CAP}_{r,p,t,v}}{UC_{r,t}}$$

temoa.extensions.unit_commitment.components.commitment.uc_started_upper_tightening_constraint(*model: UnitCommitmentModel*, *r: Region*, *p: Period*, *s: Season*, *d: TimeOfDay*, *t: Technology*, *v: Vintage*) → ExprLike [source]

Tightening: cannot start more units than are currently offline. Redundant but makes the integer formulation a bit more efficient.

$$\mathbf{UCST}_{r,p,s,d,t,v} \leq \frac{\mathbf{CAP}_{r,p,t,v}}{UC_{r,t}} - \mathbf{UCN}_{r,p,s,d,t,v}$$

temoa.extensions.unit_commitment.components.commitment.uc_stopped_upper_tightening_constraint(*model: UnitCommitmentModel*, *r: Region*, *p: Period*, *s: Season*, *d: TimeOfDay*, *t: Technology*, *v: Vintage*) → ExprLike [source]

Tightening: cannot shutdown more units than are currently online. Redundant but makes the integer formulation a bit more efficient.

$$\mathbf{UCSP}_{r,p,s,d,t,v} \leq \mathbf{UCN}_{r,p,s,d,t,v}$$

Commitment Transition

temoa.extensions.unit_commitment.components.commitment.uc_transition_constraint(*model: UnitCommitmentModel*, *r: Region*, *p: Period*, *s: Season*, *d: TimeOfDay*, *t: Technology*, *v: Vintage*) → ExprLike [source]

Commitment transition: change in online units equals started minus stopped. The constraint is indexed by the current slice (s, d) and links to its successor (s_{next}, d_{next}) via `model.time_next`.

$$\mathbf{UCN}_{r,p,s_{next},d_{next},t,v} - \mathbf{UCN}_{r,p,s,d,t,v} = \mathbf{UCST}_{r,p,s,d,t,v} - \mathbf{UCSP}_{r,p,s,d,t,v}$$

Output Bounds

temoa.extensions.unit_commitment.components.commitment.uc_min_output_constraint(*model: UnitCommitmentModel*, *r: Region*, *p: Period*, *s: Season*, *d: TimeOfDay*, *t: Technology*, *v: Vintage*) → ExprLike [source]

Output must be at least min_output_fraction * unit_capacity per online unit, converted to energy units via capacity_to_activity and segment_fraction.

$$\sum_{i,o} \mathbf{FO}_{r,p,s,d,i,t,v,o} \geq \text{minFrac} \cdot UC_{r,t} \cdot C2A_{r,t} \cdot SEG_{s,d} \cdot \mathbf{UCN}_{r,p,s,d,t,v}$$

temoa.extensions.unit_commitment.components.commitment.**uc_max_output_constraint**(*model: UnitCommitmentModel, r: Region, p: Period, s: Season, d: TimeOfDay, t: Technology, v: Vintage*) → ExprLike [source]

When used, this constraint **replaces** the standard capacity constraint, where the available capacity is based on the number of online units rather than the total process capacity. The per-timeslice capacity factor $CF_{r,s,d,t,v}$ (from `capacity_factor_process` or `capacity_factor_tech`) compounds multiplicatively with `max_output_fraction` to give the effective availability ceiling:

$$\sum_{i,o} \mathrm{FO}_{r,p,s,d,i,t,v,o} \leq \underbrace{\mathrm{maxFrac} \cdot CF_{r,s,d,t,v}}_{\text{effective availability}} \cdot UC_{r,t} \cdot C2A_{r,t} \cdot SEG_{s,d} \cdot \mathbf{UCN}_{r,p,s,d,t,v}$$

Minimum Up/Down Time

temoa.extensions.unit_commitment.components.commitment.**uc_min_up_time_constraint**(*model: UnitCommitmentModel, r: Region, p: Period, s: Season, d: TimeOfDay, t: Technology, v: Vintage*) → ExprLike [source]

Sum of starts over a cyclic window of length min_up_time_steps ending at (s,d) must not exceed online units at (s,d).

$$\sum_{\tau \in W^{\mathrm{up}}_{s,d}} \mathbf{UCST}_{r,p,\tau,t,v} \leq \mathbf{UCN}_{r,p,s,d,t,v}$$

temoa.extensions.unit_commitment.components.commitment.**uc_min_down_time_constraint**(*model: UnitCommitmentModel, r: Region, p: Period, s: Season, d: TimeOfDay, t: Technology, v: Vintage*) → ExprLike [source]

Sum of stops over a cyclic window of length min_down_time_steps ending at (s,d) must not exceed offline units at (s,d).

$$\sum_{\tau \in W^{\mathrm{dn}}_{s,d}} \mathbf{UCSP}_{r,p,\tau,t,v} \leq \frac{\mathbf{CAP}_{r,p,t,v}}{UC_{r,t}} - \mathbf{UCN}_{r,p,s,d,t,v}$$

> ⚠ Warning
>
> **Floating-point precision of time-slice hours.** The `min_x_time_hours` values are integers, but the hour length of each time-of-day segment is stored as a float. When walking backwards through the time-slice sequence to build the look-back window $W^{\mathrm{dn}}_{s,d}$, the cumulative elapsed hours are compared against the integer target with a tolerance of 10^{-6} hours. This buffer is intentionally tiny and will not mask genuine misalignment, and means that the look-back cyclic window is only correct when the segment hours are near-exact fractions of the day. If your `time_of_day` *hours* or `time_season` *segment_fraction* data are too rounded, the window boundary may be mis-classified by one time slice.

Ramp Constraints

When a technology appears in both the `unit_commitment` table and the `ramp_up_hourly` / `ramp_down_hourly` tables, the standard core-model ramp constraints (`ramp_up_constraint`, `ramp_down_constraint`) are **replaced** by UC-aware versions for that technology. Non-UC technologies retain the original core-model ramp constraints unchanged.

The core ramp constraint bounds the change in hourly activity against a fraction of total installed capacity. The UC version replaces this with a commitment-aware expression: only online units contribute to the ramp envelope, and the operating point assumed for units that start or stop during the transition adds or removes headroom according to the effective minimum output fraction. See `_ramp_constraint()` for the full mathematical derivation.

temoa.extensions.unit_commitment.components.commitment.uc_ramp_up_constraint(*model: UnitCommitmentModel, r: Region, p: Period, s: Season, d: TimeOfDay, t: Technology, v: Vintage*) → ExprLike [source]

UC-aware ramp-up constraint. Replaces the core `ramp_up_constraint` for technologies that appear in both the `unit_commitment` and `ramp_up_hourly` tables.

The ramp envelope is driven by *online units* rather than total installed capacity. Units that start during slice (s, d) arrive part-way to their maximum output, providing additional headroom; units that stop reduce it:

$$\frac{\sum_{I,O} \mathbf{FO}_{r,p,s_{next},d_{next},i,t,v,o}}{H_{s_{next},d_{next}}} - \frac{\sum_{I,O} \mathbf{FO}_{r,p,s,d,i,t,v,o}}{H_{s,d}} \leq [RH_{r,t} \cdot \Delta H \cdot \mathbf{UCN}_{r,p,s,d,t,v} + f_{\text{eff}}^{\min} \cdot \mathbf{UCST}_{r,p,s,d,t,v} - f \tag{1}$$

where $f_{\text{eff}}^{\min} = \max(\underline{f}_{r,t}, RH_{r,t} \cdot \Delta H)$ is the effective minimum output fraction for started/stopped units, and $\Delta H = (H_{s,d} + H_{s_{next},d_{next}})/2$.

temoa.extensions.unit_commitment.components.commitment.uc_ramp_down_constraint(*model: UnitCommitmentModel, r: Region, p: Period, s: Season, d: TimeOfDay, t: Technology, v: Vintage*) → ExprLike [source]

UC-aware ramp-down constraint. Replaces the core `ramp_down_constraint` for technologies that appear in both the `unit_commitment` and `ramp_down_hourly` tables.

Symmetric to `uc_ramp_up_constraint()` with signs on $\mathbf{UCST}$ and $\mathbf{UCSP}$ reversed: a unit that stops during (s, d) adds headroom (output falls), while a starting unit reduces it:

$$\frac{\sum_{I,O} \mathbf{FO}_{r,p,s,d,i,t,v,o}}{H_{s,d}} - \frac{\sum_{I,O} \mathbf{FO}_{r,p,s_{next},d_{next},i,t,v,o}}{H_{s_{next},d_{next}}} \leq [RD_{r,t} \cdot \Delta H \cdot \mathbf{UCN}_{r,p,s,d,t,v} + f_{\text{eff}}^{\min} \cdot \mathbf{UCSP}_{r,p,s,d,t,v} - f \tag{2}$$

where $f_{\text{eff}}^{\min} = \max(\underline{f}_{r,t}, RD_{r,t} \cdot \Delta H)$.

temoa.extensions.unit_commitment.components.commitment._ramp_constraint(*model: UnitCommitmentModel, ramp_up: bool, r: Region, p: Period, s: Season, d: TimeOfDay, s_next: Season, d_next: TimeOfDay, t: Technology, v: Vintage*) → ExprLike [source]

Objective Contributions

temoa.extensions.unit_commitment.components.startup.append_startup_costs(*model: UnitCommitmentModel*) → None [source]

Append discounted startup costs and startup emission costs to the total-cost objective.

Two cost terms are added:

1. **Direct startup cost**: for each (r, t) in `uc_startup_cost`, the cost per timeslice is $started\_units \cdot UC_{r,t} \cdot startup\_cost\_per\_capacity$, discounted and summed over all periods and vintages.

2. **Startup emission cost**: for each (r, p, e) in `cost_emission`, the startup emissions (from `uc_startup_emissions`) are multiplied by the emission cost rate, then discounted and added to the objective.

Output Table

Results are written to the `output_unit_commitment` table with one row per $(scenario, region, period, season, tod, tech, vintage)$:

Column	Description
`online_cap`	capacity online at the start of the timeslice ($\mathbf{UCN} \cdot UC_{r,t}$)
`start_cap`	capacity started during the timeslice ($\mathbf{UCST} \cdot UC_{r,t}$)
`stop_cap`	capacity stopped during the timeslice ($\mathbf{UCSP} \cdot UC_{r,t}$)

> **Note**
>
> **Starts and stops take effect in the following time slice.** The transition constraint links $\mathbf{UCST}_{s,d}$ and $\mathbf{UCSP}_{s,d}$ to the *change* in online count between (s, d) and its successor (s_{next}, d_{next}). A unit started during slice (s, d) therefore first appears as online — and is first able to produce output — in (s_{next}, d_{next}). Equivalently, a unit stopped during (s, d) is still counted as online for that slice and goes offline from (s_{next}, d_{next}) onward.