A Layered Sedimentation Basin in OpenHydroQual

Building a four-layer settling column with four particle classes

Abstract

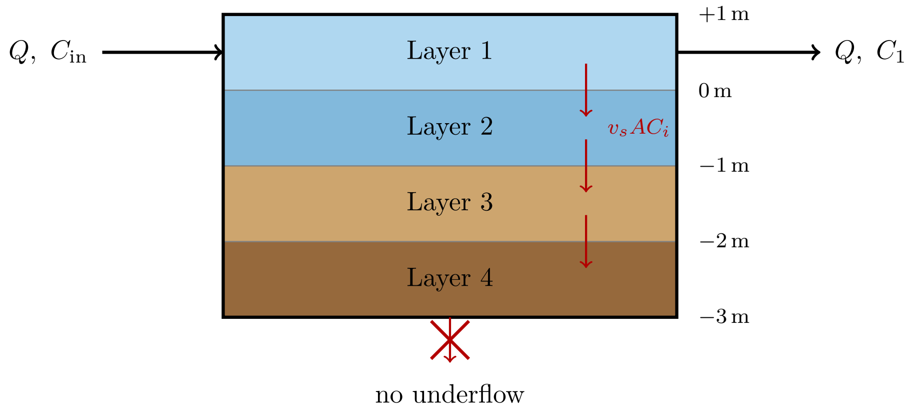
This tutorial builds one complete model from scratch: a sedimentation basin discretized into four settling layers, fed by a constant influent and drained by an equal effluent, with *no* underflow, and carrying four particle classes of differing settling velocity. It is the model shipped as `Examples/Sedimentation_Basin_Particle_Classes`. Because nothing is withdrawn from the floor, the only downward transport is gravity settling, and the model reduces to a textbook result — removal depends solely on the ratio of settling velocity to overflow rate — which makes it a good check that the tool is doing what you think it is doing. Both routes are covered: the graphical interface, and the equivalent model script. The final section compares the simulated answer with the analytical one.

Contents

1 What we are building

1.1 The physical system

A rectangular or circular basin of plan area A receives a flow Q carrying suspended solids. The same flow Q leaves as clarified effluent from the top. The basin is divided vertically into four completely mixed layers of equal depth. Solids settle from each layer into the one below at a velocity v_s that depends on the particle class. Nothing is pumped out of the bottom, so solids accumulate on the floor.



1.2 Design point

Symbol	Quantity	Value
Q	Inflow = outflow	2000 m^3/day
A	Plan area	400 m^2
$v^* = Q/A$	**Overflow rate**	**5 m/day**
z_0	Floor elevation	-3 m
H	Depth of the basin	4 m
V_{tot}	Total initial volume	1600 m^3
C_{in}	Influent concentration, per class	50 g/m^3
Q_u	Underflow	0
	Simulated period	day 36526 $\to$ 36556

Only the floor elevation and the total depth are entered. The layer bottoms are derived — each layer is $H/4 = 1$ m thick, so they come out at -3, -2, -1 and 0 m — and the total volume is divided equally, giving 400 m^3 per layer.

The four particle classes are chosen to straddle the overflow rate, so that removal spans nearly the full range from negligible to almost complete:

Class	v_s (m/day)	v_s/v^*	Expected removal
`P1`	0.5	0.1	9.1%
`P2`	2	0.4	28.6%
`P3`	10	2	66.7%
`P4`	50	10	90.9%

1.3 Components used

Role	Type	Plugin file
Particle classes	`Particle`	`main_components.json`
Influent boundary	`Reactor`	`mass_transfer.json`
The basin	`Clarifier (4 layers)`	`wastewater.json`
Effluent boundary	`fixed_head`	`main_components.json`
Influent connection	`Fixed flow`	`wastewater.json`
Effluent connection	`Effluent flow`	`wastewater.json`

`Clarifier (4 layers)` is a *composite*: dropping one on the diagram creates four `Settling compartment` blocks and the three `Sedimentation tank element interface` links between them, all driven by the composite's own properties. See `docs/composite_blocks.tex` for how composite types are declared.

2 Route A — building it in the graphical interface

2.1 Start a project and load the plugins

1. File → New Project. `main_components.json` is loaded automatically; the other two must be added.
2. File → Preferences → Add plugin opens the Select Plugin dialog, which lists the plugins that ship with OpenHydroQual. Choose Mass Transfer and press OK.
3. Open the same dialog again and choose Wastewater processes.

Both plugins are in the shipped list, so there is no need to hunt for the files on disk. Each entry in the dialog loads exactly one plugin file — Mass Transfer loads `mass_transfer.json` and Wastewater processes loads `wastewater.json` — which is why the dialog is opened twice. (File → Preferences → Add plugin from file does the same job through a file browser, and is only needed for a plugin you wrote yourself and that is not in the list.)

The Add Objects menu and the toolbars rebuild themselves each time a plugin is added. Menu entries show each type's *description*, not its type name — so the basin appears as Clarifier: sedimentation basin discretized into four settling layers, and composites are listed after a separator at the bottom of Add Objects → Blocks.

2.2 Declare the four particle classes

Constituents must exist *before* the blocks, so that every block picks up the per-constituent properties.

1. Add Objects → Chemistry → Solid Particle.
2. In the property panel set Name to `P1` and Settling velocity to `0.5` m/day.
3. Repeat three more times for `P2`, `P3` and `P4` with 2, 10 and 50 m/day.

Leave Turbidity Coefficient and Turbidity Exponent at zero; they only affect the reported `turbidity` output.

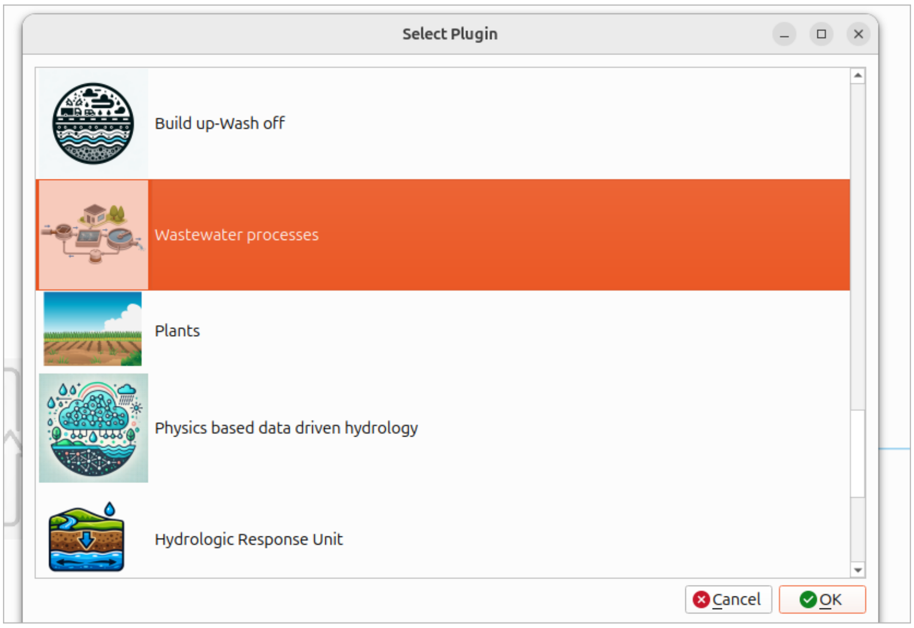


Figure 1: The Select Plugin dialog, reached through File → Preferences → Add plugin, with Wastewater processes selected. Mass Transfer is loaded the same way.

Table 1: Properties of the `Influent` reactor.

Property	Meaning	Value
Name	–	`Influent`
Initial Storage	working volume	200 m^3
Bottom elevation	–	0 m
Constant inflow	external feed	2000 m^3/day
P1..P4 Concentration	initial condition	50 g/m^3 each
P1..P4 Constant Inflow concentration	feed strength	50 g/m^3 each

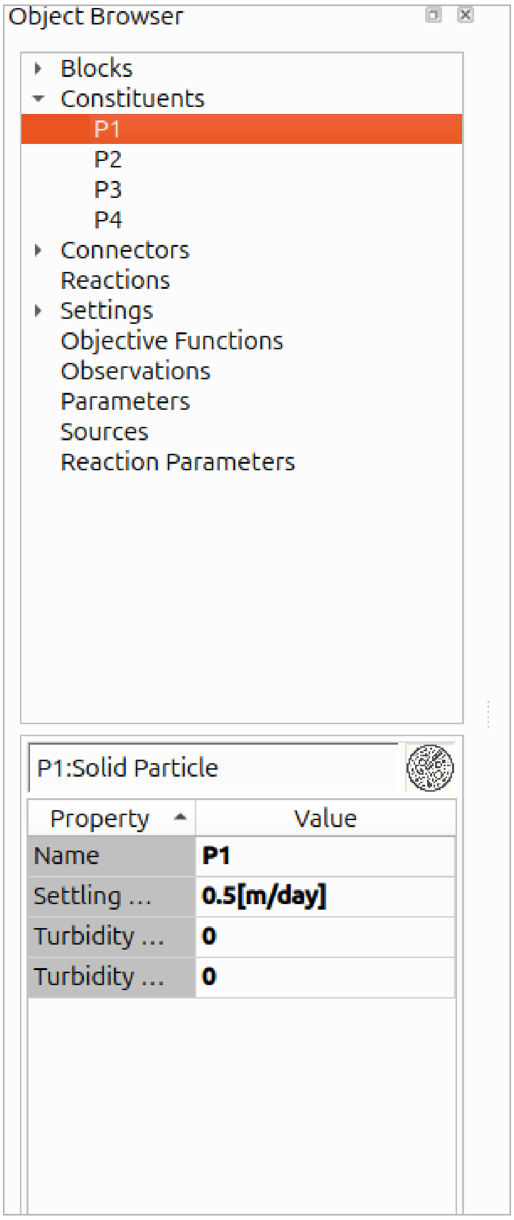


Figure 2: The four particle classes under Constituents in the Object Browser, with `P1` selected and its settling velocity set to 0.5 m/day.

2.3 Place the influent boundary

1. Add Objects → Blocks → Reactor, and click on the canvas to place it.
2. Set the properties in Table 1.

This block is a boundary condition, not part of the basin. It receives 2000 m^3/day of water at 50 g/m^3 per class and passes the same flow on, so it holds a constant concentration and acts as an inexhaustible source. Setting both the initial *and* the inflow concentration to 50 starts it at steady state, avoiding a startup transient.

2.4 Place the basin

Add Objects → Blocks, then the entry Clarifier: sedimentation basin discretized into four settling layers below the separator. One click creates the whole four-layer stack. Set:

Table 2: Properties of the `Sed_Basin` composite. This handful of values describes the whole four-layer stack; the per-layer geometry is derived from them. The two optional time-series inputs, External inflow time series into the top layer and Time variable component of the underflow, are left empty here.

Property	Meaning	Value
Name	–	`Sed_Basin`
Cross-sectional area of the layer interfaces	A	400 m^2
Flow drawn downward through the layers	Q_u	**0**
Elevation of the clarifier floor	z_0	-3 m
Depth of the clarifier	H	4 m
Initial volume of water in the clarifier	V_{tot}	1600 m^3

Three points deserve emphasis.

The underflow is what makes this a settling column rather than a clarifier.
Setting Flow drawn downward through the layers to zero puts zero advective flow on all three internal interfaces at once. The only remaining downward transport is settling.

The layer geometry is derived, not entered.
The four layers are equal quarters of the depth, so their bottom elevations follow from the floor and the depth alone:

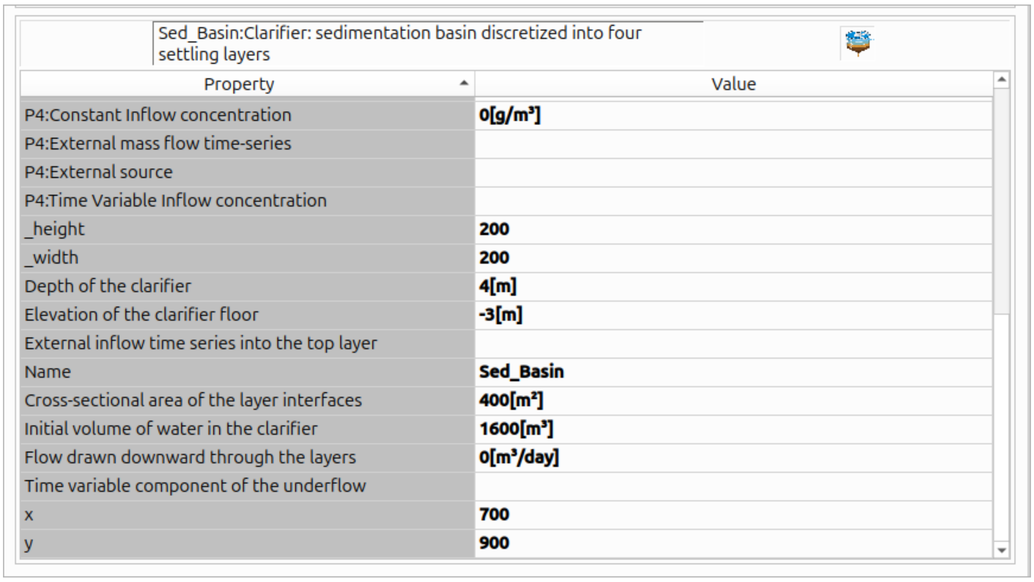
$$z_4 = z_0, \qquad z_3 = z_0 + \frac{H}{4}, \qquad z_2 = z_0 + \frac{H}{2}, \qquad z_1 = z_0 + \frac{3H}{4}, \tag{1}$$

which for this design point gives -3, -2, -1 and 0 m. Likewise V_{tot} is split four ways, giving 400 m^3 per layer. This is what keeps the stack self-consistent: because the elevations are generated rather than typed, they are always strictly decreasing downward, and the settling gate of Section 4 can never be accidentally closed.

Volume and depth are independent.
V_{tot} is an *initial condition* — how much water is in the basin at $t = 0$ — while H and A describe the vessel. For a full prismatic basin they agree, $V_{tot} = A\,H = 1600$ m^3, as here, but nothing forces that.

2.5 Place the effluent boundary

Add Objects → Blocks → Fixed-head boundary. Name it `Effluent`, set Storage to 100000, Bottom elevation to 0 and head to 0. It is a receiving water body large enough not to matter.



Sed_Basin:Clarifier: sedimentation basin discretized into four settling layers

Property	Value
P4:Constant Inflow concentration	0[g/m³]
P4:External mass flow time-series	
P4:External source	
P4:Time Variable Inflow concentration	
_height	200
_width	200
Depth of the clarifier	4[m]
Elevation of the clarifier floor	-3[m]
External inflow time series into the top layer	
Name	Sed_Basin
Cross-sectional area of the layer interfaces	400[m²]
Initial volume of water in the clarifier	1600[m³]
Flow drawn downward through the layers	0[m³/day]
Time variable component of the underflow	
x	700
y	900

Figure 3: The property panel of the `Clarifier (4 layers)` composite. Every value of Table 2 is visible at once — note Depth of the clarifier and Elevation of the clarifier floor in place of four separate layer elevations, Initial volume of water in the clarifier in place of four layer volumes, and Flow drawn downward through the layers $= 0$. There is no per-layer entry anywhere in the panel: that geometry is derived. The `P4:...` rows at the top are the per-constituent properties, which stay at zero — the basin starts clean.

2.6 Draw the two links

Link types are toggles: pick the type, then drag from the source block to the destination block.

1. Add Objects → Links → Fixed flow. Drag from `Influent` to `Sed_Basin`. Name it `Influent to Basin` and set Flow rate to 2000 m^3/day.
2. Add Objects → Links → Effluent flow (clarified water drawn from the top of a clarifier). Drag from `Sed_Basin` to `Effluent`. Name it `Basin to Effluent` and set Flow rate to 2000 m^3/day — the *same* number.

You never pick a layer. The composite resolves the endpoint itself from its `external_links` table: the wildcard entry sends any unlisted link type to layer 1, and `Effluent flow` is mapped explicitly to layer 1 as well. Had you attached a `Sludge flow` link, it would have landed on layer 4 — which is exactly what we are *not* doing here.

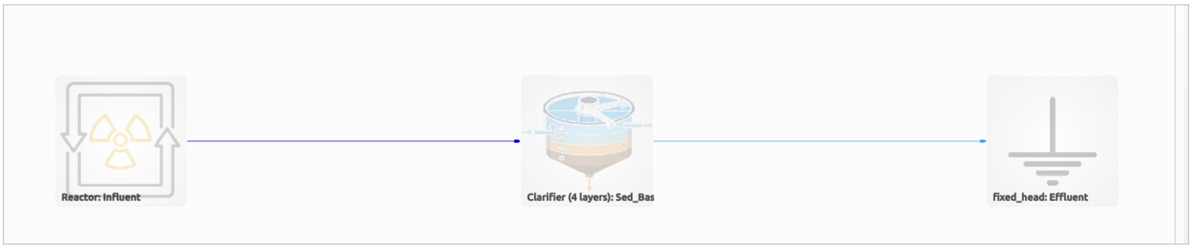


Figure 4: The finished diagram: three nodes and two links. The four layers live inside the single `Clarifier (4 layers)` node.

2.7 Simulation settings

Open View → Object Browser, expand Settings and select general_parameters:

- Simulation start time = 36526
- Simulation end time = 36556 (30 days)
- Output File Name (All outputs) = `output.txt`

The defaults under solver_settings are fine; this model is small, linear in the constituents, and converges without tuning.

2.8 Validate, run, save

Model → Validate, then Model → Run. Then File → Save as to write the `.ohq` model script.

2.9 Viewing results

The engine never simulates a composite — it simulates the members — so the results live on the layers, and the basin's context menu groups them one submenu per member. Right-click the basin on the diagram and follow

Results → Layer1 → P1:Concentration

to plot the effluent concentration of the first particle class. The member submenus are labelled by role — Layer1 through Layer4 — rather than by the instance-qualified names (`Sed_Basin__Layer1`) that appear in `output.txt`. Walking the same menu with a different layer or a different class gives every combination:

- Layer1 → P*n*:Concentration is the effluent quality for class n. It should settle at the value in Section 5.
- Layer2 and Layer3 plotted for the same class should level off at the *same* value as layer 1 — the middle layers convey rather than accumulate.
- Layer4 → P*n*:Concentration rises linearly and never levels off. That is the sludge blanket, and it is the expected behaviour of a basin that is never desludged.

The three internal interfaces are on the same menu, below the layers. They are worth a look:

- interface12 → Flow rate should be flat zero, which is the direct confirmation that there is no underflow.
- interface12 → P*n*:Settling transfer rate is the settling flux in g/day. Compared against the same quantity on interface34, the two converge to the same number once steady — the conveyor again.

If a plot looks blank or flat at an implausible value, check the axis range before suspecting the model: the plot window keeps whatever zoom it was last given, and a series can sit entirely outside the visible region.

3 Route B — the model script

An `.ohq` file is a plain-text command script, so the whole model can be written by hand. The listing below is complete and runnable; it produces the results in Section 5. Values carry no unit suffix, which means they are in the engine's base units — metres, grams and *days*.

```
loadtemplate; filename = <path>/resources/main_components.json
addtemplate;  filename = <path>/resources/mass_transfer.json
addtemplate;  filename = <path>/resources/wastewater.json

setvalue; object=system, quantity=simulation_start_time, value=36526
setvalue; object=system, quantity=simulation_end_time,   value=36556
setvalue; object=system, quantity=alloutputfile,         value=output.txt

create constituent;type=Particle,name=P1,settling_velocity=0.5
create constituent;type=Particle,name=P2,settling_velocity=2
create constituent;type=Particle,name=P3,settling_velocity=10
create constituent;type=Particle,name=P4,settling_velocity=50

create block;type=Reactor,name=Influent,Storage=200,bottom_elevation=0,
    constant_inflow=2000,
    P1:concentration=50,P2:concentration=50,
    P3:concentration=50,P4:concentration=50,
    P1:constant_inflow_concentration=50,P2:constant_inflow_concentration=50,
    P3:constant_inflow_concentration=50,P4:constant_inflow_concentration=50,
    x=0,y=900

create composite;type=Clarifier (4 layers),name=Sed_Basin,
    surface_area=400,underflow=0,
    floor_elevation=-3,depth=4,total_volume=1600,
    x=700,y=900

create block;type=fixed_head,name=Effluent,Storage=100000,
    bottom_elevation=0,head=0,x=1400,y=900

create link;from=Influent,to=Sed_Basin,type=Fixed flow,
    flow=2000,name=Influent to Basin
create link;from=Sed_Basin,to=Effluent,type=Effluent flow,
    flow=2000,name=Basin to Effluent
```

Each `create` command must occupy a single line — the line breaks above are for the page only. A few rules the parser enforces:

- Commands run in order, and `create constituent` must precede the blocks, otherwise the blocks never acquire `P1:...` properties.
- Type names may contain spaces and parentheses (`Clarifier (4 layers)`); only `;`, `,` and `=` are structural.
- A value may carry an explicit unit in brackets, as `settling_velocity=0.5[m/day]`. Without brackets the value is interpreted in base units, which for a velocity is m/day.
- Whitespace around `=` and `,` is trimmed.

What the interface saves is the same script with every property written out explicitly, including defaults — see `Examples/Sedimentation_Basin_Particle_Classes/Sedimentation_Basin_Particle_Classes.ohq`.

4 What the model actually computes

4.1 Mass transfer across a link

Every `Particle` contributes a `masstransfer` expression to each link:

```
"masstransfer": "_ups(flow;concentration)
   + settling_velocity*area*( _hsd(bottom_elevation.s-bottom_elevation.e-1e-7)*concentration.s
                            - _hsd(bottom_elevation.e-bottom_elevation.s-1e-6)*concentration.e
    )"
```

The first term is ordinary advection, upstream-weighted. The second is settling: `_hsd` is a Heaviside step, so the term is active only in the direction of decreasing bottom elevation, and `area` is the interface area. For an interface link between layer i (above) and layer $i+1$ (below) this is simply

$$J_{i\to i+1} \;=\; v_s\,A\,C_i. \tag{2}$$

This is also why the `Fixed flow`, `Effluent flow` and `Sludge flow` link types declare an `area` of 10^{-9}: they are plumbing, and must not settle anything.

4.2 Steady state

With $Q_u = 0$, only layer 1 has throughflow. Its mass balance at steady state is

$$Q\,C_{\text{in}} \;=\; \underbrace{Q\,C_1}_{\text{effluent}} \;+\; \underbrace{v_s A\,C_1}_{\text{settled}} \qquad\Longrightarrow\qquad \boxed{C_1 \;=\; \frac{C_{\text{in}}}{1+v_s A/Q}} \tag{3}$$

so the removal efficiency depends on nothing but the ratio of settling velocity to overflow rate $v^* = Q/A$:

$$\eta \;=\; 1-\frac{C_1}{C_{\text{in}}} \;=\; \frac{v_s/v^*}{1+v_s/v^*}. \tag{4}$$

Layers 2 and 3 receive $v_s A C_{i-1}$ and pass on $v_s A C_i$, so at steady state $C_2 = C_3 = C_1$: they are a conveyor, not a sink. Layer 4 has an inlet and no outlet, so it accumulates at a constant rate and its concentration grows linearly,

$$C_4(t) \;=\; \frac{Q\,C_{\text{in}}\,\eta}{V}\;t, \tag{5}$$

which is the sludge blanket. This is the physically correct behaviour of a basin that is never desludged, and it is the one quantity in the model that never reaches steady state.

5 Verification

Running the script above for 30 days and reading `Sed_Basin__Layer1_Pn:concentration` from `output.txt` reproduces the analytical solution to five significant figures:

Class	v_s (m/day)	C_1 simulated	C_1 analytical	C_4 at day 30	Removal
`P1`	0.5	45.4545	45.4545	609	9.1%
`P2`	2	35.7143	35.7143	2131	28.6%
`P3`	10	16.6667	16.6667	5119	66.7%
`P4`	50	4.5454	4.5455	7028	90.9%

Three further checks are worth running on any model of this kind:

- C_2 and C_3 equal C_1, confirming the middle layers are conveying rather than accumulating.
- The interface flows `Sed_Basin__interface12_flow` and its two siblings are exactly zero, confirming there is no underflow.
- Mass balance closes on each class. For `P3`: 100 000 g/day in = 33 333 g/day out + 66 667 g/day settled.

6 Pitfalls

Read layer 1, not the effluent block.

Effluent quality is `Sed_Basin__Layer1_Pn:concentration`. The column `Effluent_Pn:concentration` is the concentration inside the 100 000 m^3 receiving reservoir, which is still filling and is much lower. This trips people up on the first run.

Equal elevations silently disable settling.

The settling term is gated on $z_i > z_{i+1}$: give two adjacent layers the same bottom elevation and solids simply stop moving, with no diagnostic. The `Clarifier (4 layers)` composite forecloses this by deriving the elevations from the floor and the depth, but the trap is live in any stack assembled from bare `Settling compartment` blocks — including one produced by Ungroup. If such a model shows zero removal, check the elevations first.

Constituents before blocks.

In a hand-written script, a `create block` that precedes `create constituent` produces a block with no particle properties, and the `Pn:...` assignments are silently dropped.

Watch the log for missing-property errors.

Messages of the form `property 'X' does not exist in 'Y'` mean an expression referenced a quantity that is not there, and the expression quietly evaluated to zero. They look like noise and are not. The one remaining message in this model, `property 'inflow' does not exist in 'Effluent'`, is benign — a `fixed_head` boundary has no inflow term, so its loading is correctly zero.

The console executable needs composite support compiled in.

`terminal/TOpenHydroQual/` must include `Composite.cpp`, otherwise `create composite` is ignored and the run fails with `Block 'Sed_Basin' does not exist!`.

7 Variations to try

- **Turn it into a real clarifier.** Attach a `Sludge flow` link from the basin to a waste boundary and set Flow drawn downward through the layers to that rate. The link lands on layer 4 automatically, and layer 4 reaches a steady sludge blanket instead of accumulating.
- **Change the overflow rate.** Halving Cross-sectional area of the layer interfaces to 200 m^2 doubles v^* to 10 m/day and pushes every class's removal down; the tabulated η values move to 4.8, 16.7, 50 and 83.3%.
- **Deepen the basin.** Raise Depth of the clarifier on its own and the four layer elevations move together, staying evenly spaced. Steady-state removal does not budge — it depends on Q/A, not on depth — but the transient does, through the layer volumes.
- **Refine the discretization.** Compare against the two-compartment `Clarifier`. With no underflow the steady-state effluent is identical — it depends only on layer 1 — but the sludge blanket and the transient differ. Note that the two-compartment type still takes its interface and floor elevations individually; only `Clarifier (4 layers)` derives them.
- **Use a measured distribution.** Replace the four classes with a settling-velocity distribution from a column test; the model structure is unchanged.