OpenHydroQual Tutorial 1

Foundations: Blocks, Links, Sources, and the Mass Balance

Arash Massoudieh* Behzad Shakouri†

June 1, 2026

Overview

OpenHydroQual is an open-source platform for building and simulating water-quantity and water-quality models. You assemble a model from a small set of object types — *blocks*, *links*, and *sources* — and the solver integrates the resulting system of mass-balance equations through time.

This first tutorial introduces those building blocks by constructing the simplest meaningful model: a single catchment that receives rainfall and drains to a downstream boundary.

Learning objectives. After completing this tutorial you will be able to:

- explain what a block, a link, and a source represent;
- read and interpret the storage mass balance that the solver integrates;
- place objects, set their properties, and connect them in the GUI;
- run a simulation and inspect the results.

What you will build. A *Catchment* block forced by a *Precipitation* source, draining through a *Catchment link* into a *Fixed-head* boundary.

Prerequisites. A working OpenHydroQual installation. No prior modeling experience is assumed. All objects used here belong to the core (*main components*) set, so no plugins are required.

Key Concepts

Blocks and the mass balance

A **block** is a control volume that stores something — typically water. Every block carries a state variable called *Storage*, and the solver advances it in time by enforcing a mass balance: the rate of change of storage equals everything coming in minus everything going out. For a block i,

$$\frac{dS_i}{dt} = \underbrace{I_i(t)}_{\text{direct inflow}} + \underbrace{\sum_k q_{k,i}^{\text{src}}}_{\text{sources}} - \underbrace{\sum_{l \in \mathcal{L}_i} Q_l}_{\text{net flow through links}} , \tag{1}$$

*EnviroInformatics
†The Catholic University of America

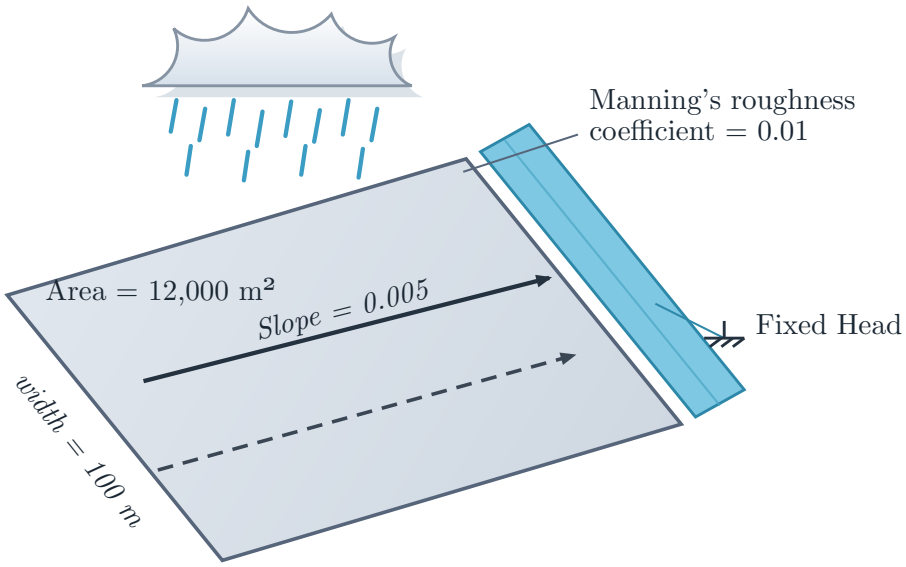

Figure 1: Conceptual model built in this tutorial: a single catchment receives rainfall from a precipitation source and drains down-slope through a link into a fixed-head boundary representing the receiving stream. The labelled quantities (area, width, slope, and Manning's roughness coefficient) are the catchment properties assigned in Step 2.

where S_i is the storage, I_i a user-supplied inflow time series, $q^{\text{src}}_{k,i}$ the contribution of each attached source, and Q_l the flow carried by each link connected to the block. Derived quantities such as water depth and hydraulic head are computed from S_i through block-specific expressions (for a catchment, depth = Storage/area).

Links (connectors)

A **link** carries flow between two blocks — a start block (`.s`) and an end block (`.e`). The flow is given by an expression that usually depends on the hydraulic heads of the two blocks it connects, for example a gradient-driven form

$$Q_l = f(h_{\text{s}},\, h_{\text{e}},\, \boldsymbol{\theta}_l)\,, \tag{2}$$

where $\boldsymbol{\theta}_l$ collects the link's parameters (length, width, roughness, etc.). The same Q_l enters eq. (1) as an outflow for the start block and an inflow for the end block, which is what conserves mass across the system.

Sources

A **source** is a reusable object that injects (or removes) water or mass according to a rule or a time series — precipitation and evapotranspiration are the canonical examples. A source is created once and then *attached* to one or more blocks, where it appears as the q^{src} term in eq. (1). This is distinct from a block's direct *inflow* time series: a source is a shared, named object; an inflow is a property of a single block.

The project file and the solver

A model is stored as a single `.ohq` file. When you run it, the solver discretizes time and solves the coupled mass-balance equations (eq. (1)) at each step, iterating with a Newton–Raphson scheme

because the link flows in eq. (2) generally depend nonlinearly on the unknown heads. You do not need to configure any of this to follow the tutorial; it is described here only so the results make sense.

1 Step 1: Your First Block

The most basic object in any OpenHydroQual model is a *block*. In this step we add a single *catchment* block and take a tour of its properties. We will not change any values yet; the goal is to understand what a block exposes and how the property panel works before we build a model that actually does something in Steps 2 and 3.

1. **Start OpenHydroQual.** You are presented with an empty workspace. On the left is the *Object Browser*, which organizes every object in the model into branches — **Blocks**, **Connectors**, **Constituents**, **Sources**, **Settings**, and so on. It is empty for now because we have not added anything.

2. **Add a catchment block.** On the **Blocks** toolbar, click the catchment icon to place a catchment in the workspace. A green catchment block labelled *Catchment: Catchment (1)* appears, and a corresponding entry is added under the **Blocks** branch of the Object Browser (fig. 2).

3. **Open the property panel.** Click the block once to select it. The panel in the lower-left now lists all of the catchment's properties in two columns, *Property* and *Value*.

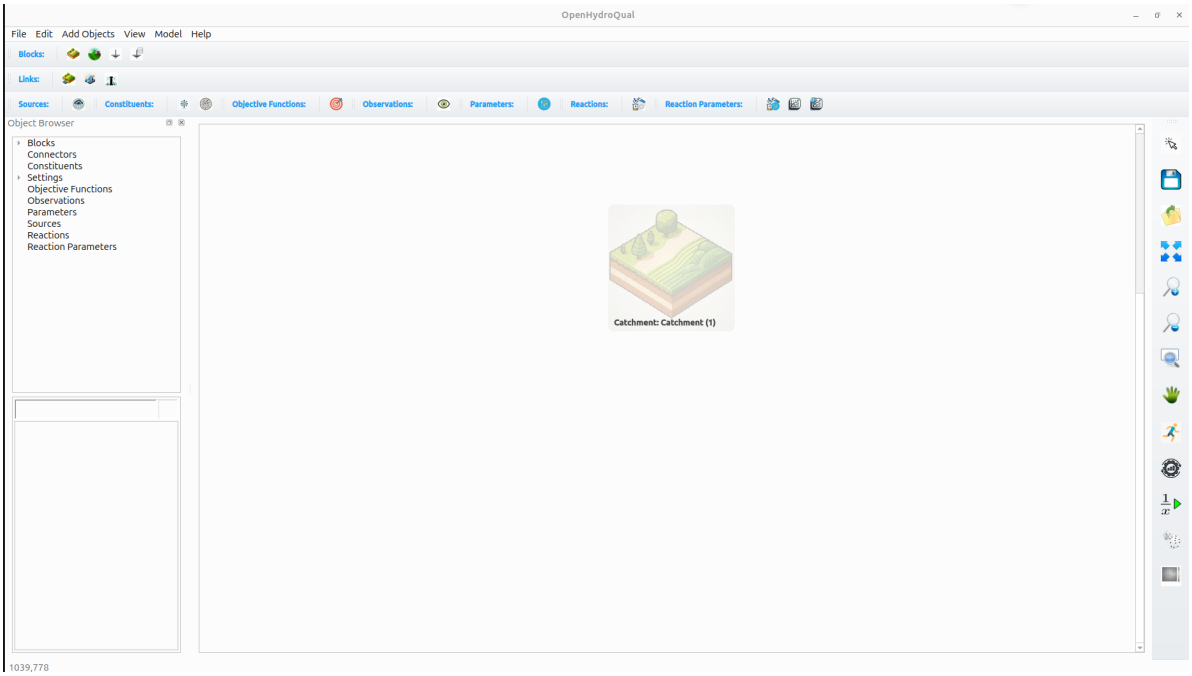

Figure 2: The workspace after adding a single catchment block. The block appears on the canvas and under the **Blocks** branch of the Object Browser; its properties are listed in the panel below.

A tour of the catchment's properties. Scrolling through the panel reveals that even this single block exposes many properties. They fall into three broad groups:

- **Forcing and attachments** — *Precipitation*, *Evapotranspiration*, and *Inflow*. The first two are slots to which you attach *source* objects (the q^{src} term in eq. (1)); *Inflow* accepts a direct inflow time series (the I_i term). They are empty by default, meaning the catchment receives no water until we provide some in Step 2.
- **Hydrologic and geometric properties** — *Catchment area*, *Catchment slope in the direction of flow*, *The width of the catchment*, *Manning's Roughness Coefficient*, *Runoff coefficient*, *Depression storage*, *Mean elevation of the catchment*, *Average water depth* (the initial depth), and *Loss coefficient*. These describe the physical catchment and govern how stored water is converted to outflow.
- **Bookkeeping and display** — *Name*, the canvas position (x, y), and the on-screen icon size (`_height`, `_width`). These affect only how the block is identified and drawn; they have no effect on the simulation.

Units are selectable. Wherever a property has physical dimensions, its value is shown with the current unit in brackets — for example `0[m^2]` for the area or `0[1/day]` for the loss coefficient. The unit is not fixed: click into the value cell and a unit drop-down appears next to it, letting you enter the quantity in whatever unit is most convenient. OpenHydroQual converts to its internal units automatically, so you never have to convert by hand. Throughout these tutorials, when we give a value such as "area = 12,000" you may enter it in any of the offered units as long as the number matches the unit you select.

Accepting the defaults. Many of these properties already hold sensible default values (for instance the runoff coefficient defaults to 1), and a real study would set the physical ones to match the catchment being modelled — as the dedicated rainfall–runoff tutorial does. For this introductory model, however, *we leave every value at its default and change nothing.* Our purpose here is simply to understand the anatomy of a block: that it has a storage governed by eq. (1), a set of properties with selectable units, and empty forcing slots waiting to be filled. We fill the first of those slots in the next step.

2 Step 2: Adding a Source

The catchment we placed in Step 1 has empty forcing slots, so nothing drives it yet. In this step we create a *precipitation source*, give it real rainfall data, attach it to the catchment, and assign the catchment's physical properties. We then run the model and watch its storage respond. Attaching the source activates the source term q^{src} in the mass balance of eq. (1).

The sample rainfall data. We will use a sample precipitation record covering 2010–2012. Download it from:

`https://openhydroqual.com/wp-content/Data/Sample_Rain_Data%20%28mm%29.csv`

and save it somewhere convenient on your computer. The file is a plain comma-separated table; opening it in a text editor shows rows like:

```
40179,40179.02,0
40186.0416666667,40186.0833333333,0.254
40186.0833333333,40186.125,0.254
...
```

Understanding the structure. Each row has *three* columns:

(1) the **start time** of an interval,

(2) the **end time** of that interval, and

(3) the **precipitation depth** that fell during the interval.

The two time columns are dates expressed as *serial day numbers*: a single number counting days (and fractions of a day) from a fixed origin, which is the same convention spreadsheet programs use for dates. In this file day 40179 corresponds to 1 January 2010 at midnight, and the record runs to late December 2012. A fractional part is the time of day — for example 40186.0417 is 08 January 2010 at about 01:00. Most intervals here are one hour long.

Representing the data as start/end pairs rather than single time stamps makes the series unambiguous: each value is explicitly tied to the interval over which it was measured, so OpenHydroQual knows exactly how to distribute the rainfall in time when it integrates eq. (1).

What the unit means. The third column is a precipitation *depth* for the interval, here in millimetres (the file is the "mm" version). A depth of rainfall over a known area and a known interval is what the model needs: internally, OpenHydroQual combines the depth with the interval length to obtain a rate, and multiplies by the catchment area to obtain a volumetric inflow (the source's *coefficient* is the catchment area). The precipitation unit is selectable on the value box — and, importantly, its *default is metres*, so for this file we must change it to millimetres to match the data.

Creating the source and loading the data.

1. **Create a precipitation source.** On the **Sources** toolbar, click the precipitation icon . A new *Precipitation (1)* entry appears under the **Sources** branch of the Object Browser. Select it to show its properties.

2. **Load the data file.** In the property panel, click the value box next to the *Precipitation Cumulative* property and select the `Sample_Rain_Data (mm).csv` file you downloaded. Its path now appears in the value box.

3. **Set the unit to millimetres.** On the same value box, use the unit selector to change the time-series unit from its default of *metres* to *mm*, so it matches the units in the file. This step is essential: leaving it at metres would over-count the rainfall by a factor of one thousand.

4. **Plot to verify.** Right-click the value and choose **Plot** from the context menu that appears. A plot window opens showing the rainfall hyetograph spanning 2010–2012, confirming the file was read correctly (fig. 3). You can switch the plot's own *Unit* selector and *Display* style at the top of that window.

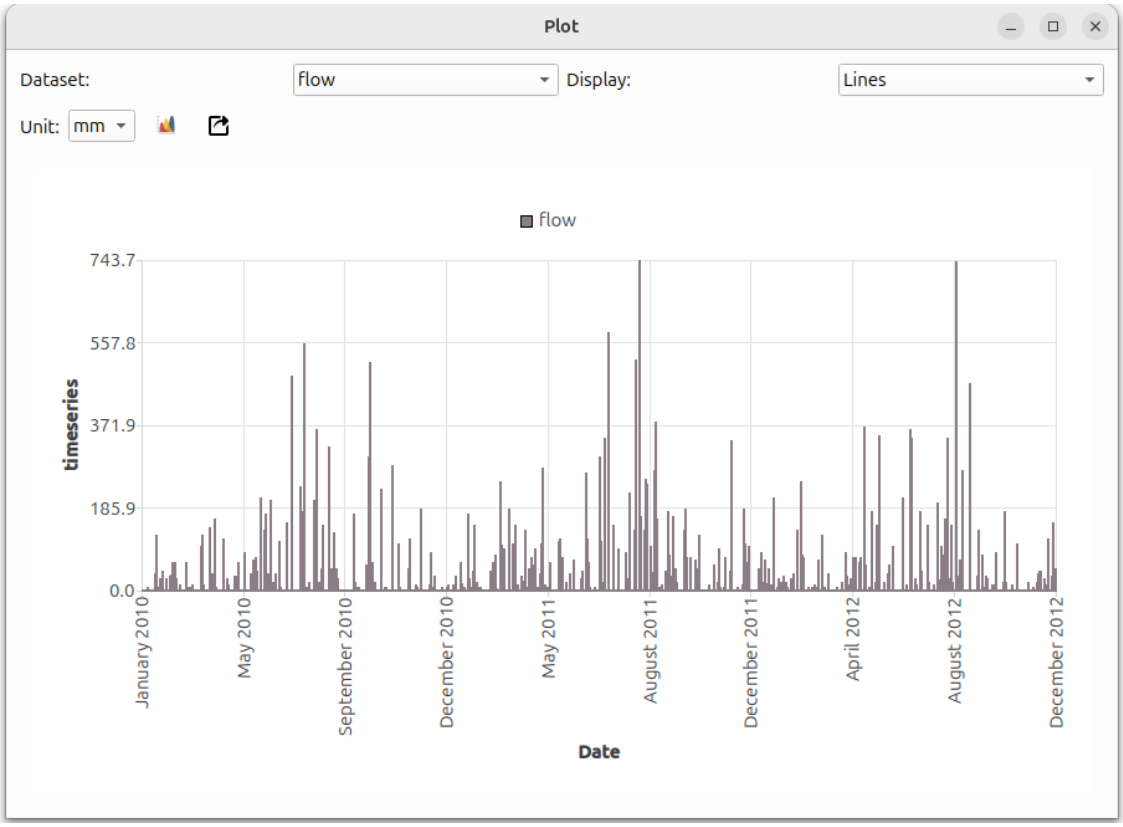

Figure 3: The sample precipitation series after loading, plotted through the right-click **Plot** menu to verify it spans the intended 2010–2012 period. The plot window's *Unit* selector is set to mm.

Parameterizing the catchment. A source is a stand-alone object until it is *attached* to a block, and a catchment cannot be simulated until its physical properties are set. We now do both on the catchment we created in Step 1.

1. **Select** the catchment block.
2. **Attach the precipitation source.** In the catchment's property panel, find the *Precipitation* slot (one of the empty forcing slots we noted in Step 1) and choose *Precipitation (1)* from the drop-down.
3. **Assign the required physical properties.** Unlike Step 1, where we accepted the defaults, the catchment now needs the following properties to be set for the model to run. These values correspond to the conceptual model in fig. 1; recall that every dimensional value may be entered in any of the offered units, shown here in brackets.

Property	**Value**
Catchment area	12,000 [m^2]
Manning's Roughness Coefficient	0.01
The width of the catchment	100 [m]
Catchment slope in the direction of flow	0.005

4. **Runoff coefficient.** This property defaults to 1, meaning all precipitation reaches the catchment's storage. You may leave it at 1 or change it; we keep it at 1 here.

The area, Manning's coefficient, width, and slope must be assigned for the model to run. Of these, the area takes immediate effect — it scales the precipitation into a volumetric inflow — while the roughness, width, and slope govern how quickly water leaves the catchment, an effect that becomes visible once we add an outlet in Step 3. Note also the distinction the program makes between a *source* (a named, reusable object that can be attached to several blocks at once) and a block's own *Inflow* property (a time series belonging to that single block).

Setting the simulation period. Before running, tell the solver what time window to simulate. In the Object Browser, open **Settings** → **General Settings**. The property panel (headed *General Settings: General Parameters*) lists the *Simulation start time* and *Simulation end time*, among other options (fig. 4). Click each value box and pick the dates so the run covers a one-year period:

Setting	**Value**
Simulation start time	1/1/2010
Simulation end time	1/1/2011

The solver only integrates over this window, so precipitation outside it is ignored. Although the sample file extends through 2012, simulating a single year keeps this first run short; you can extend the end time later to see the longer-term behaviour.

Running and observing the storage. Save the model and run it with the *Run Model* button . When the run finishes, right-click the catchment and choose **Results** → **Storage** to open the plot. Because the catchment has rainfall coming in but no outlet yet, its storage only ever increases: it steps up during each rain event and holds steady between them, climbing from zero to roughly 10,000 m^3 over the simulated year (fig. 5). The derived *water depth* (Storage divided by area) rises in the same way. This is the source term q^{src} of eq. (1) acting on its own. In the next step we add an outlet so that water can also leave, completing the balance.

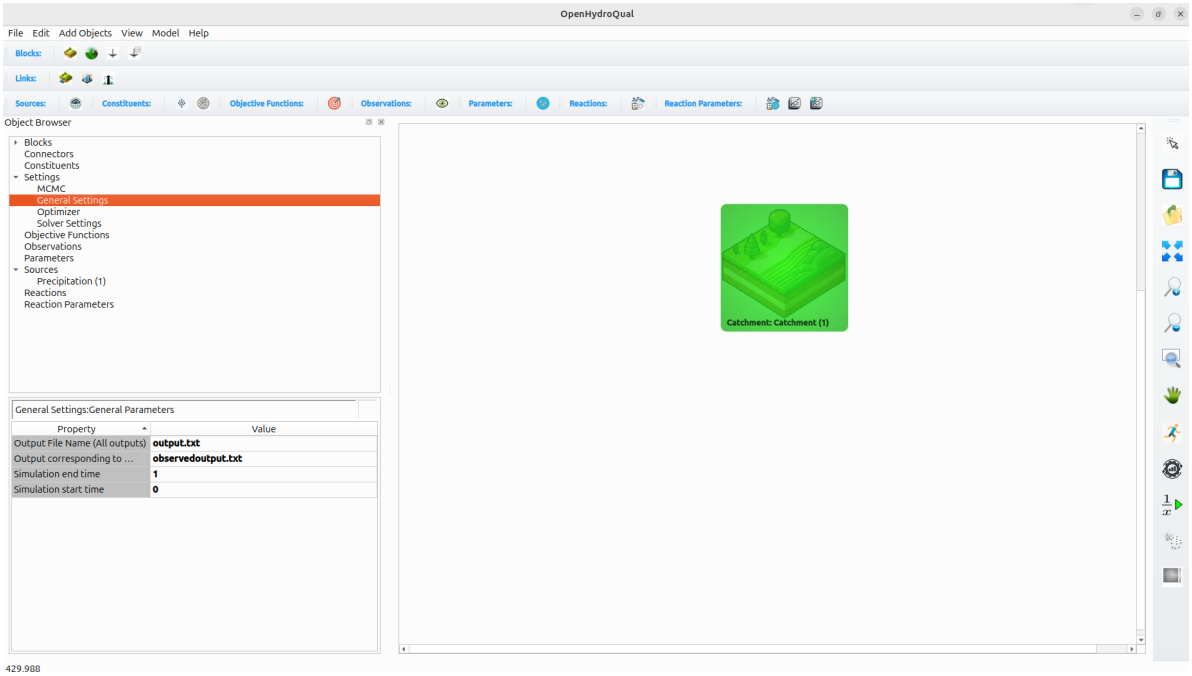

Figure 4: Setting the simulation period under **Settings** → **General Settings**. Pick the *Simulation start time* and *Simulation end time* to span 1/1/2010–1/1/2011.

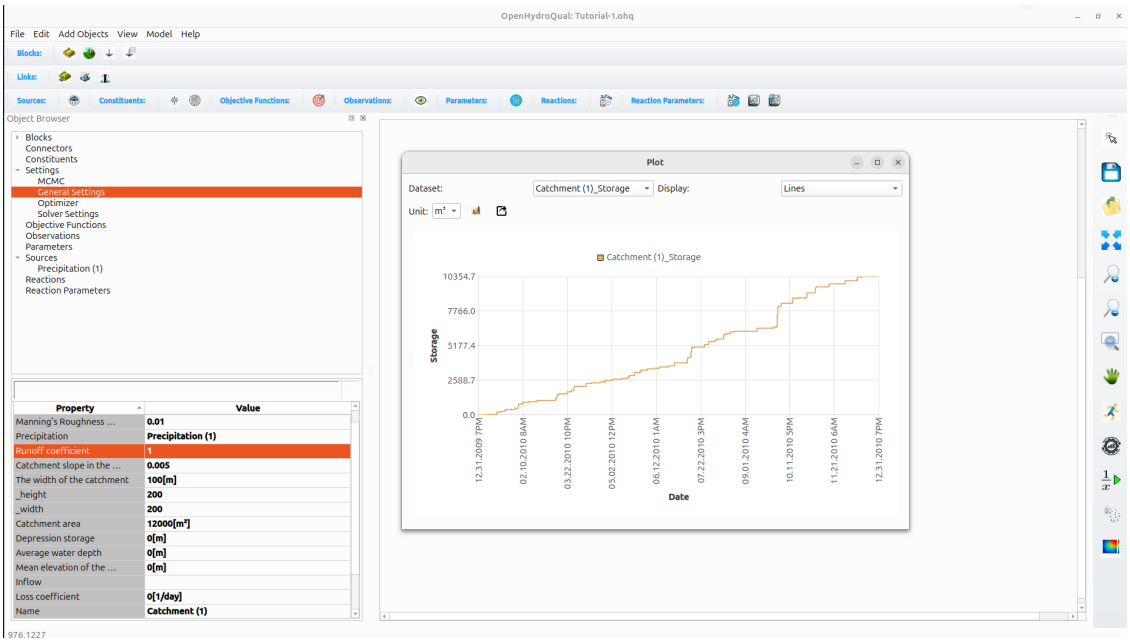

Figure 5: Catchment storage over 2010, obtained by right-clicking the catchment and choosing **Results** → **Storage**. With rainfall but no outlet, storage accumulates monotonically, stepping up at each rain event.

Tip: the built-in component reference

Before we connect anything, it is worth knowing where to look up what each object does. From the top menu, choose **Help → Component Description** to open the *Object Type Reference* (fig. 6). The left-hand list contains every object type available in the program — blocks, connectors, sources, reactions, settings, and so on — and a *Filter* box at the top lets you jump to one by name.

Selecting an object shows its *Description*, its *Object Type*, the number of properties it exposes, and a full *Properties* table. For each property the table gives its name, its type (*Value*, *Expression*, *Source*, *Balance*, and so on), a short description, and a *Details* column. The Details column is especially useful: for computed quantities it shows the actual governing expression. For the catchment, for instance, it reveals that the effective precipitation is `Precipitation*Runoff_coeff` and that the precipitation loss is `(0-Precipitation*(1-Runoff_coeff))` — which is exactly why a runoff coefficient of 1 (our setting) sends all rainfall to storage and leaves no loss.

This reference is the authoritative source for the meaning of every property you encounter in these tutorials. Whenever a property is unclear, or you want to see the equation a block or link actually solves, open this dialog rather than guessing. We will rely on it again as we introduce new components.

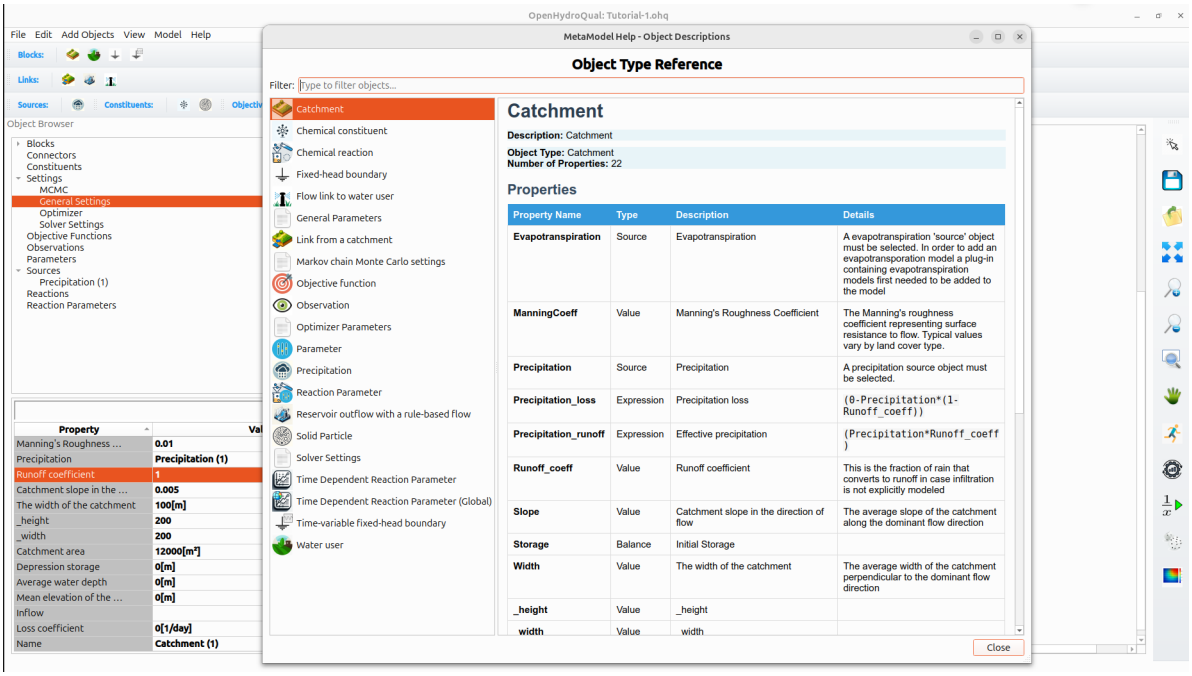

Figure 6: The *Object Type Reference*, opened from **Help → Component Description**. Selecting the catchment lists all of its properties together with their types, descriptions, and — in the *Details* column — the governing expressions.

3 Step 3: Adding a Connector

So far water only accumulates, because the catchment has no outlet. In this step we give it one: a downstream boundary and a connector that carries water away. This activates the outflow term $\sum_l Q_l$ in the mass balance of eq. (1), so that storage can both rise (from rainfall) and fall (through the outlet).

1. **Add a fixed-head boundary.** On the **Blocks** toolbar, click the fixed-head button to place a *Fixed-head boundary* on the canvas. This block represents the receiving stream into which the catchment drains. Select it and set its head:

Property	Value
Head	0 [m]

   A fixed-head boundary holds its water-surface elevation (its *head*) at a value you prescribe, no matter how much water flows into or out of it — it is an infinite reservoir at a fixed level. It is the standard way to represent the edge of a model: a stream, lake, or far-field boundary whose level we treat as known. Here a head of 0 places the receiving stream at the model datum.

2. **Add the connector.** On the **Links** toolbar, click the *Link from a catchment* connector, then drag from the catchment (the start block) to the fixed-head boundary (the end block). An arrow appears pointing from the catchment to the boundary, showing the positive flow direction (fig. 7).

3. **Review the link's properties.** Select the new link. Its property panel shows only a *Name* (defaulting to *Catchment (1) - fixed_head (1)*); there is nothing else to set. This is deliberate: the overland flow this link carries is computed from the *catchment's* own width, Manning's roughness coefficient, slope, and water depth — the very properties we assigned in Step 2 — so the link needs no parameters of its own. You can confirm the governing expression in the component reference (**Help** → **Component Description**, then *Link from a catchment*).

4. **Run the model.** Save the model and run it. A *Running Model* window opens and reports progress as the solver works through the simulation period (fig. 8). When it reaches 100% it prints *Finished!* and a log of what it did — writing the output files, saving the model state, and so on — including the folder where the results were saved.

   The plot in this window, *Time-step size vs Time*, is worth a glance. OpenHydroQual does not march through time in fixed steps; it adapts the step size as it goes, taking small steps when conditions change rapidly (during a rain event, when storage and flow are shifting quickly) and larger steps when things are calm. This is the adaptive Newton–Raphson solver described in the *Key Concepts* section at work: the spiky trace simply shows the solver shortening its steps whenever the solution becomes harder to track. You do not need to act on it, but a trace that collapses to very small steps and never recovers is a useful warning sign that a model is struggling to converge.

5. **Inspect the results.** Right-click the link and choose **Results** → **Flow** to see the discharge to the stream (fig. 9), and right-click the catchment for its **Storage**. The flow now tracks the rainfall as a series of runoff pulses — sharp peaks during storms, falling back toward zero between them — rather than the ever-rising storage of Step 2. Storage likewise rises during rain events and recedes between them as water drains through the link. What enters as rainfall now leaves as outflow: the complete mass balance of eq. (1) in action.

A note on the flow direction and the boundary head. The link's flow is positive from its start block to its end block — here, from the catchment to the stream. For this overland-flow connector the discharge is set by the catchment's depth and slope rather than by the difference in

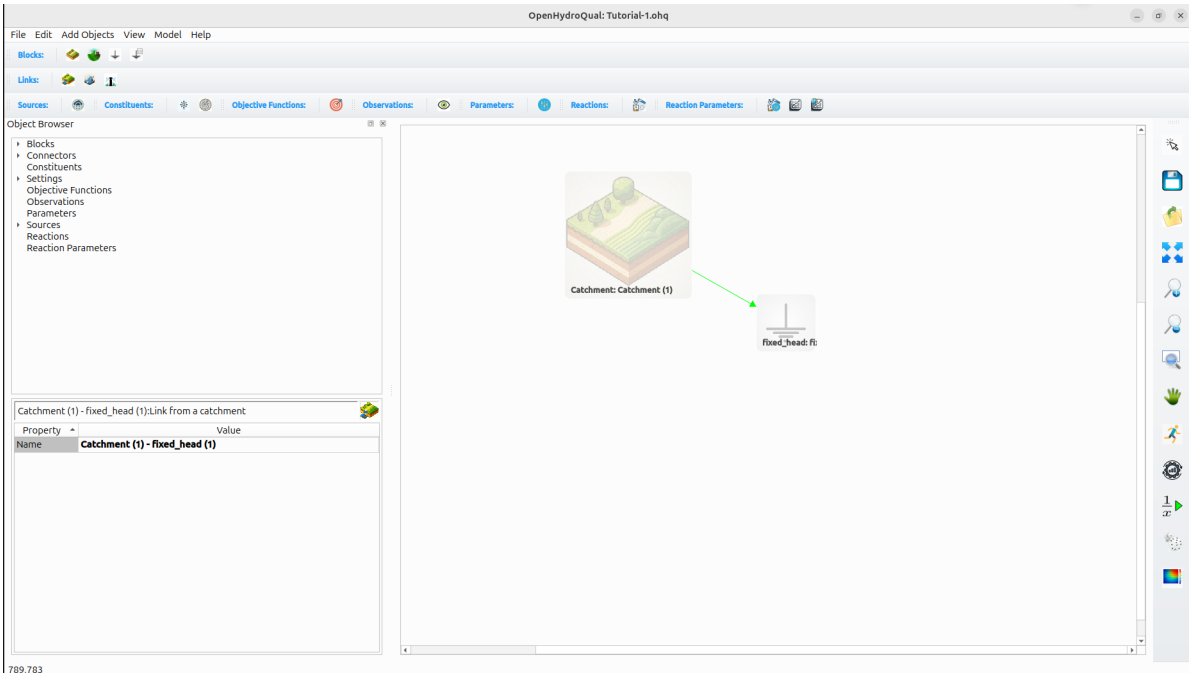

Figure 7: The completed model: the catchment (forced by the precipitation source) drains through a *Link from a catchment* into the fixed-head boundary that represents the receiving stream. The selected link exposes only a *Name*, since its flow is governed by the catchment's own properties.

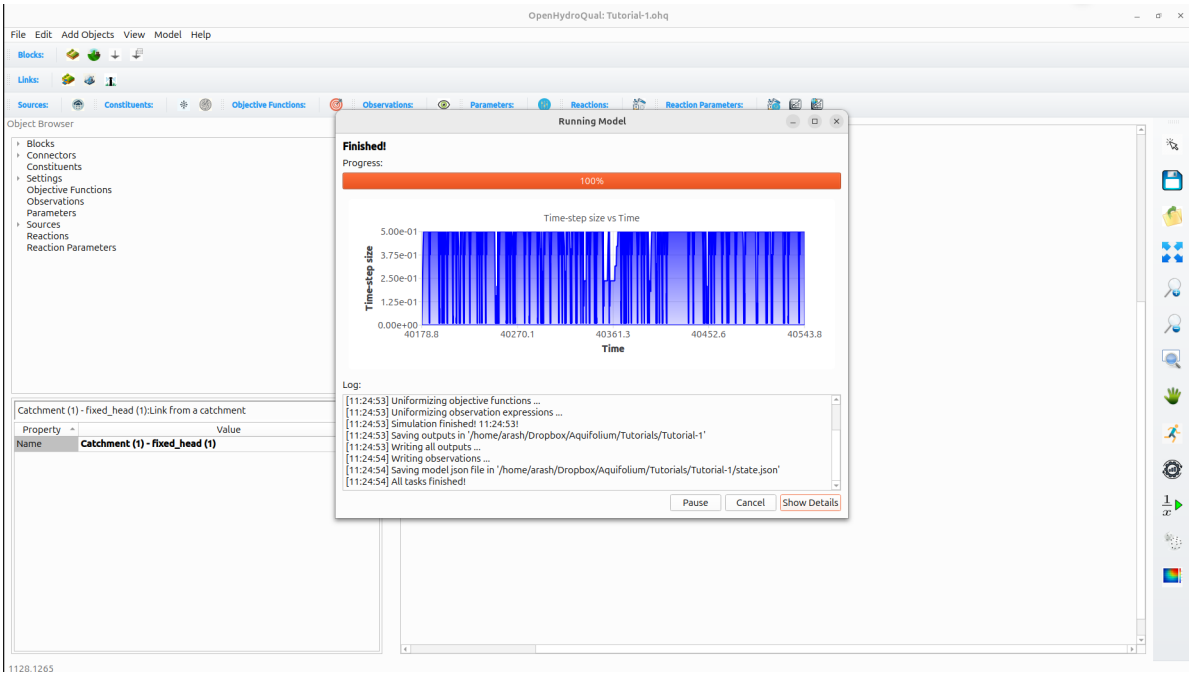

Figure 8: The *Running Model* window after a completed run. The progress bar reads 100%, the log records the output and state files that were written, and the *Time-step size vs Time* plot shows the adaptive solver varying its step size through the simulation.

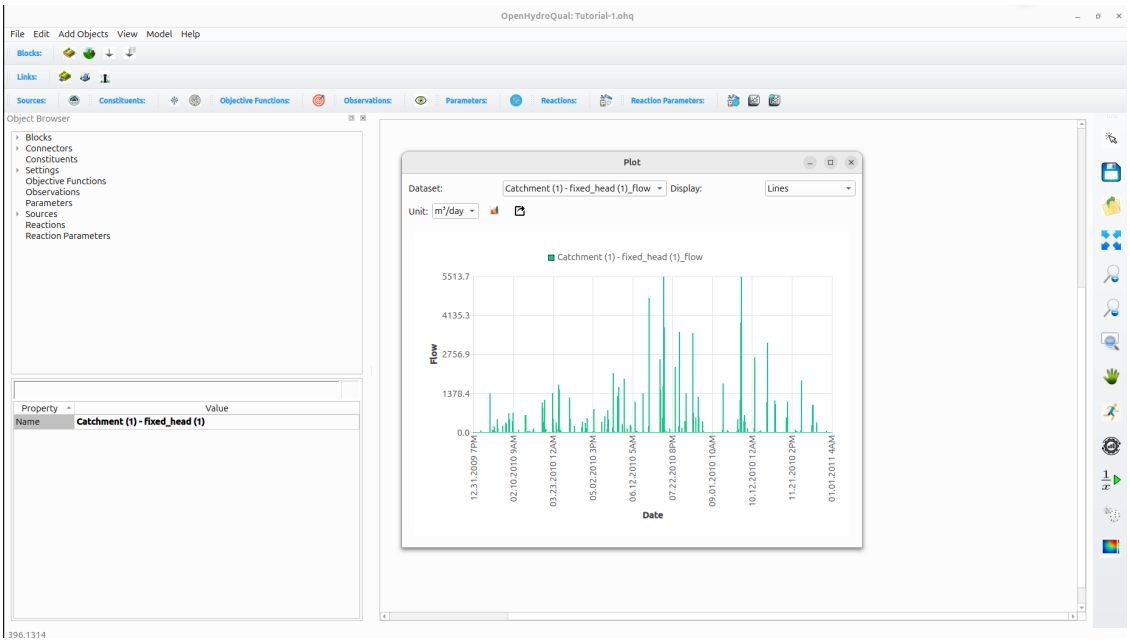

Figure 9: Flow from the catchment into the fixed-head boundary over 2010, obtained by right-clicking the link and choosing **Results** → **Flow**. Each peak is a runoff response to a rain event; between events the flow recedes toward zero.

head across the link, so the fixed-head value you chose defines where the downstream boundary sits without throttling the outflow. Many other connectors in OpenHydroQual (pipes, groundwater links, weirs) instead drive flow by the head difference $h_s - h_e$ between the blocks they join; in those cases the fixed-head value directly shapes the flow. We will meet head-driven links in later tutorials.

4 Interpreting the Output

- **Plotting.** Right-click any object and use the **Results** menu to plot a state or derived quantity. Multiple series can be overlaid and formatted with distinct colors or markers.
- **What to look at.** For this model: catchment **Storage** and **depth**, and the link **Flow**. Confirm that what enters (rain) leaves (link flow) once storage stabilizes — this is eq. (1) in action.

5 Anatomy of the `.ohq` File

When you save a model, OpenHydroQual writes a single file with the `.ohq` extension. It is worth opening this file in a plain text editor at least once, because it is not an opaque binary blob: it is a human-readable *command script*. Each line is an instruction, and replaying the lines from top to bottom reconstructs the model exactly. This is why `.ohq` files are easy to track in version control, compare with a colleague, or even edit by hand.

The file we just built contains four kinds of lines.

Loading the component definitions. The first lines tell OpenHydroQual which component templates the model relies on:

```
loadtemplate; filename = .../resources/main_components.json
addtemplate; filename = .../resources/main_components.json
```

These point to the JSON files that define each object type — what properties a catchment has, how a link computes its flow, and so on. `loadtemplate` starts a fresh template set and `addtemplate` adds to it; a model that used a plug-in (a pond, a soil column) would list its template here too. Everything that follows refers back to the types defined in these files.

System and solver settings. A long block of `setvalue` lines records every model-wide setting:

```
setvalue; object=system, quantity=simulation_start_time, value=40178.8
setvalue; object=system, quantity=simulation_end_time, value=40543.8
setvalue; object=system, quantity=initial_time_step, value=0.01
setvalue; object=system, quantity=minimum_timestep, value=1e-06
setvalue; object=system, quantity=nr_tolerance, value=0.001
...
```

Here you can see the simulation period we set in Step 2, stored as the serial day numbers 40178.8 and 40543.8 (1/1/2010 and 1/1/2011) — the same date convention as the precipitation file. The remaining entries are the defaults behind the **General Settings**, **Solver Settings**, **Optimizer**, and **MCMC** branches of the Object Browser: the Newton–Raphson tolerance and time-step limits that produced the adaptive step trace in the run window, together with settings for the genetic-algorithm optimizer and the MCMC sampler that later tutorials will use. We left all of these at their defaults.

Creating the objects. The final lines create the model itself — and these are the ones you will recognize, because they hold the exact values you entered:

```
create source; type=Precipitation, name=Precipitation (1),
    timeseries=.../Sample_Rain_Data (mm).csv [mm]

create block; type=Catchment, name=Catchment (1),
    Precipitation=Precipitation (1), ManningCoeff=0.01,
    Runoff_coeff=1, Slope=0.005, Width=100, area=12000, ...

create block; type=fixed_head, name=fixed_head (1), head=0,
    Storage=100000, ...

create link; type=Catchment_link, name=Catchment (1) - fixed_head (1),
    from=Catchment (1), to=fixed_head (1)
```

Each line names an object's *type*, gives it a *name*, and lists its properties as `property=value` pairs. A few details are worth noting:

- The precipitation source records both the path to the data file and the unit we chose, `[mm]` — exactly the unit selection from Step 2.
- The catchment line carries the four properties we assigned (`ManningCoeff`, `Slope`, `Width`, `area`) and the `Precipitation=Precipitation (1)` attachment, alongside the defaults we left untouched. Empty entries such as `Evapotranspiration=` and `inflow=` are the forcing slots we never filled.

- The link line simply records its endpoints (`from` the catchment, `to` the fixed head) and its type; as we saw in Step 3, it stores no flow parameters of its own.

Why this matters. Because the file is an ordered list of commands, the order is meaningful: templates load first, then settings, then objects — and a source must be created before the block that references it, just as the catchment here is created after `Precipitation (1)`. You rarely need to edit an `.ohq` file by hand, since the GUI writes it for you, but being able to read it makes models transparent: you can see at a glance exactly what a saved model contains, diff two versions to find what changed, or troubleshoot a value that does not look right.

6 The Output Files

Running the model produces several files, written to the folder reported in the run window's log (the same folder as the `.ohq` file). At this stage most of them relate to calibration and optimization, which this simple model does not use, so we introduce them only briefly here and return to them in the calibration tutorial. Two, however, are worth knowing now.

`output.txt` — **the simulation results.** This is the main results file: a comma-separated table of every quantity flagged for output, recorded through time over the simulation period. It holds the same time series you viewed interactively by right-clicking objects, in a plain-text form you can load into a spreadsheet or plotting tool. Its name is set by the *Output File Name (All outputs)* property under **Settings → General Settings**, which is why it appeared as `output.txt` in the `.ohq` file.

The layout has one feature worth pointing out. Rather than a single shared time column, *each recorded quantity is preceded by its own time column*, so the columns come in `t, value` pairs. The header row names them, with each value column tagged by its object and quantity:

```
t, Catchment (1)_Storage, t, Catchment (1)_depth, t,
   Catchment (1) - fixed_head (1)_flow, t, ...
40178.80000, 0, 40178.80000, 0, 40178.80000, 0, ...
40178.81000, 0, 40178.81000, 0, 40178.81000, 0, ...
...
40544.19000, 0.00700, 40544.19000, 5.83e-07, 40544.19000, 0.00249, ...
```

This paired-column form exists because each series can be reported on its own set of time points (the adaptive solver does not place every quantity's samples at identical instants). The times are serial day numbers, the same convention as the input data, running here from 40178.8 to about 40544.2. For our model the recorded quantities include the catchment's storage, depth, head, precipitation, runoff, and loss; the fixed head's storage and head; the link's flow; and the precipitation intensity — every quantity whose definition was flagged for output.

`state.json` — **the saved model state.** At the end of a run OpenHydroQual writes a JSON snapshot of the model's state, noted in the run log ("Saving model json file…"). It captures the model and the condition it reached, which the program can use to resume or restart from where the run finished rather than from the initial condition.

`errors.txt` — **model-structure and run errors.** This file logs problems detected in the model itself — errors in the model structure or in evaluating it during the run, such as an expression that

cannot be computed or a property left in an invalid state. It is a diagnostic log, not a results file: on a clean run like this one it is empty, and if a run misbehaves it is the first place to look for what went wrong. Note that despite its name it has nothing to do with the residuals between modeled and observed values; those belong to the calibration files below.

The calibration and optimization files. The remaining files support parameter estimation and goodness-of-fit assessment, which require *observations* and an *objective function* — objects this single-catchment model does not contain. For a model with no observations they are written empty or with placeholder content, so there is nothing to inspect yet:

- `observedoutput.txt` — the modeled values sampled at the times and locations of any observations, for comparison against measured data. Because our model defines no observations, this file is empty.
- `mapped_modeled_results.txt` — modeled results aligned ("mapped") to the observation points.
- `fit_measures.txt` — summary goodness-of-fit statistics (for example R^2, Nash–Sutcliffe efficiency).
- `objective_function_values.txt` and `Objective_Function_TimeSeries.txt` — the value of each objective function, as a single number and as a time series.

We cover these in full in the calibration tutorial, where we add observations and an objective function and use them to estimate parameters. For now it is enough to know they exist and why they are empty.