Open Aquarium Project

OPEN AQUARIUM PROJECT SPECIFICATION

Dosing

PRE-DRAFT

PRE-DRAFT

Revision	0.3
Status	Pre-Draft. A candidate document submitted for consideration.
Date	2026-10-18
Identifier	dosing
Authors	Trent Scholl, Project Lead, Open Aquarium Project
Editors	Trent Scholl
License	Community Specification License 1.0

> **Pre-Draft.** This is a proposed initial draft. The Working Group has not approved it, and it will change. Comment by opening an issue with the `dosing` label.

1 License

This Specification is subject to the Community Specification License 1.0 (the "Specification License"), available at https://github.com/CommunitySpecification/1.0 and in `LICENSES/Community-Spec-1.0.md`. Contributions to this Specification are made under the Community Specification Contributor License Agreement 1.0.

The checks file, JSON Schema, and example files published with this Specification are licensed under the Apache License 2.0 (`LICENSES/Apache-2.0.txt`).

The Scope that bounds the patent commitments made under the Specification License is set out in `Scope.md`. Exclusion notices, license acceptances, and withdrawals are recorded in `Notices.md`.

Notes:

1. Appendices are informative (non-normative) unless an appendix is explicitly marked normative.

THESE MATERIALS ARE PROVIDED "AS IS." The Contributors and Licensees expressly disclaim any warranties (express, implied, or otherwise), including implied warranties of merchantability, non-infringement, fitness for a particular purpose, or title, related to the materials. The entire risk as to implementing or otherwise using the materials is assumed by the implementer and user. IN NO EVENT WILL THE CONTRIBUTORS OR LICENSEES BE LIABLE TO ANY OTHER PARTY FOR LOST PROFITS OR ANY FORM OF INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES OF ANY CHARACTER FROM ANY CAUSES OF ACTION OF ANY KIND WITH RESPECT TO THIS DELIVERABLE OR ITS GOVERNING AGREEMENT, WHETHER BASED ON BREACH OF CONTRACT, TORT (INCLUDING NEGLIGENCE), OR OTHERWISE, AND WHETHER OR NOT THE OTHER MEMBER HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

2 Compliance with Open Aquarium Tenets

2.1 Openness

The Specification is published under the Community Specification License 1.0, and its checks file, schema, and examples under the Apache License 2.0. Every safety check it defines, and every requirement, has a published test procedure (Section 19.5), so owners, makers, and reviewers can verify a product's claims for themselves. The program format is plain JSON with a published schema and validator, and the validator includes a reference implementation of dose times and of planning a day's amount from a result.

2.2 Interoperability

One program format runs the same way on a controller and on a dosing head that runs its own program (Section 15), so a doser from any maker works with any controller, and a program moves between them. Dosing is a function type of the Automation specification, so it moves between controllers with the rest of an owner's configuration. It uses Device Classes `doser` endpoints, Automation inputs, conditions, interlocks, and modes, Data Model units, observations, and events, Fail-Safe Behavior terms, and Alarms and Notifications condition codes, and adds no parallel vocabulary. Measurement-driven dosing accepts results from any analyzer, manual test, or laboratory that the Data Model can record.

2.3 Local Control

Every program runs on the controller or on the doser itself, with no network service, account, or vendor. A doser that runs its own program keeps dosing when its controller or the network is down (Section 15.2). Owners can export their programs, the saved state of each dosing day, and their dose history.

2.4 Safety

This Specification exists to meet this tenet: an overdose of alkalinity, calcium, or a trace element can kill livestock within hours, and a dosing pump that runs too long can empty a container of supplement into a system. Every head has two independent layers of limits, the program's and the head's own, which the head enforces whatever asks it to dose (Section 14). A dose that was sent is never sent again, and one that a power cut interrupts never resumes (Section 10.5); missed doses are caught up only within the day's amount (Section 10.4); and a dosing day is never dosed twice, even when the clock steps back (Section 7.5). Results change the daily amount only within the owner's limits, a step at a time, and only when they are plausible, judged once, by the testing function that screens them where there is one; when results stop, or come from a reagent long past its expiry, dosing returns to its normal amount (Section 12). A head that doses when it should not is stopped and held until a person checks it (Section 14.5). Primes and calibration runs, which add product that no dose counts, are started only by a person, and heads limit them (Section 10.6). A head that runs its own program, configured to `continue`, keeps the holds its controller sent when it loses its controller or its power, and any head keeps a hold that an interlock or an emergency mode latched (Section 15.3.4). Products that react with each other are kept apart (Section 9.6), and the parts of a multi-part supplement stay in ratio (Section 9.7).

2.5 Sustainability

Container forecasts let owners buy and refill products before they run out (Section 11.2), and product life warnings reduce waste (Section 11.5). Comparing a container's real use with the head's count shows when tubing has worn (Section 8.5), so it is replaced when it needs to be rather than on a fixed schedule. Measurement-driven dosing gives a system what it uses, rather than a generous fixed amount. Because programs are portable, owners can keep their dosers when they change controllers.

Revision History

Revision	Date	Description	Authors
0.1	2026-10-14	Initial Pre-Draft.	Trent Scholl
0.2	2026-10-16	Heads that run their own program keep `fail-safe` after power-up, keep their holds with `failSafeAction continue` and across a power cut, and never resume an interrupted dose; primes and calibration runs only when a person asks, no longer than `maxContinuousRun`, and primes within `maxDailyPrimeTime`; a dosing day never planned twice, and a clock that steps back treated as not valid; `dosedToday` reset at most once a day; a window without `end` ends at the same local time the next day; catching up limited to twice a dose's volume before catching up, off by default, and proposed for alkalinity, calcium, and magnesium supplements; adjustment limits proposed from a product's effect, and a new normal amount proposed when adjusted amounts stay far from it; the hold command, calibration points, and records of Device Classes revision 0.2 and Data Model revision 0.2, with the provisional names kept for devices of revision 0.1; a POSIX time zone derived from `timeZone` when a configuration has none; daily totals of doses by dosing day; products in records as Data Model Products; additives through a top-off left to the Water Level and Water Change specification; a test procedure, mapped to its requirements, for every requirement; references cited by tag; one sentence per line; every question for the Project Lead decided.	Trent Scholl
0.3	2026-10-18	The Project Lead's decisions of 17 October 2026 applied: the function type `dosing`, with its saved state as `state.functions["<function id>"]`, in place of the provisional names, which controllers still read; refills recorded with `record-maintenance`, and results that steer dosing entered with `enter-results`; from 30 days (provisional) after a reagent expires, no day planned from its results, but at the normal amount, recorded with the basis `expired-reagent`; a head whose results a testing function screens takes their status from it, and does not judge them again; the schema's `$id` `https://openaquariumproject.org/schemas/dosing/0.3`. Also: a hold with no known end that an interlock or an emergency mode demands latched on a head that runs its own program, and which holds outlast a failed controller; unexpected dosing never found in a head under local control, and defined for a head that runs its own program; a program of only `posixTimeZone` for a head that its controller runs, so that it knows local midnight; a controller that shows which heads wait while its clock is not valid for dosing, and asks a	Trent Scholl

Revision	Date	Description	Authors
		person to confirm the time after a step back of more than 24 hours (provisional); `maxContinuousRun`, which every head has from Device Classes revision 0.5; `planned` defined as a dose's volume before catching up; a dosing day that crosses midnight recorded as one total in documents of Data Model revision 0.3 or later; the Water Testing specification a normative reference; magazine articles no longer cited; the example description of Device Classes revision 0.5; "A pass criterion that checks what a SHOULD asks applies to a product that does it"; corrected test procedures; references to the final revisions of the other specifications; issues 16 to 21 and 23 deferred beyond revision 1.0.	

PRE-DRAFT

Contents

1 License ... 2

2 Compliance with Open Aquarium Tenets ... 2
- 2.1 Openness ... 2
- 2.2 Interoperability ... 2
- 2.3 Local Control ... 2
- 2.4 Safety ... 3
- 2.5 Sustainability ... 3

Revision History ... 3

3 Scope ... 7

4 References ... 7
- 4.1 Normative References ... 7
- 4.2 Informative References ... 8

5 Terms, Definitions, and Conventions ... 8
- 5.1 Requirements Language ... 8
- 5.2 Terms and Definitions ... 8
- 5.3 Notation ... 9

6 Introduction ... 9
- 6.1 Purpose ... 9
- 6.2 Design Principles ... 9
- 6.3 Overview ... 10
- 6.4 Example ... 11

7 Common Requirements ... 11
- 7.1 Where Programs Run ... 11
- 7.2 The Dosing Function ... 11
- 7.3 Heads ... 12
- 7.4 Volumes, Rates, and Units ... 12
- 7.5 Dosing Days and Local Time ... 12
- 7.6 Holds ... 13
- 7.7 Saved State ... 13
- 7.8 Faults and Alarms ... 14
- 7.9 Visibility ... 15

8 Calibration ... 15
- 8.1 Overview ... 15
- 8.2 Calibrating a Head ... 15
- 8.3 Several Speeds ... 15
- 8.4 When to Calibrate ... 16
- 8.5 Checking Calibration Against Containers ... 16
- 8.6 Safety Checks ... 16

9 Programs ... 17
- 9.1 Overview ... 17
- 9.2 Members ... 17
- 9.3 Dosing Days ... 18
- 9.4 Dose Times and Volumes ... 18
- 9.5 Dose Rate ... 18
- 9.6 Spacing Between Heads ... 19
- 9.7 Ratio Dosing ... 19
- 9.8 Safety Checks ... 19

10 Giving Doses ... 20
- 10.1 The Life of a Dose ... 20
- 10.2 Giving a Dose ... 20
- 10.3 Delays ... 21
- 10.4 Missed Doses and Catching Up ... 21
- 10.5 Never Twice ... 21
- 10.6 Manual Doses ... 22
- 10.7 Safety Checks ... 22

11 Containers ... 23
- 11.1 Tracking ... 23

PRE-DRAFT

11.2 Forecasts and Warnings ... 23
11.3 Empty Containers ... 23
11.4 Refills ... 23
11.5 Product Life ... 24
11.6 Safety Checks ... 24

12 Measurement-Driven Dosing ... 24
12.1 Overview ... 24
12.2 Members ... 24
12.3 Results ... 25
12.4 Plausibility ... 25
12.5 Planning the Day ... 26
12.6 Limits ... 26
12.7 Stale and Missing Results ... 26
12.8 Product Effects ... 27
12.9 Safety Checks ... 27

13 Conditions ... 28
13.1 Pausing on a Condition ... 28
13.2 Interlocks and Modes ... 28
13.3 Safety Checks ... 28

14 Safety Limits ... 29
14.1 Two Layers ... 29
14.2 Limits on the Head ... 29
14.3 Choosing Head Limits ... 29
14.4 The Day's Amount ... 29
14.5 Unexpected Dosing ... 30
14.6 Dosing Heads Only ... 30
14.7 Duty Limits ... 30
14.8 Siphoning and Placement ... 30
14.9 Safety Checks ... 31

15 Heads That Run Their Own Programs ... 31
15.1 Head Programs ... 31
15.2 Running a Head Program ... 32
15.3 Under a Controller ... 32
15.4 Size ... 33
15.5 Safety Checks ... 33

16 History ... 34
16.1 Dose Records ... 34
16.2 Adjustments ... 34
16.3 Calibrations and Refills ... 34

17 Fail-Safe Behavior ... 35
17.1 Outputs ... 35
17.2 Lost Inputs ... 35
17.3 Restarts and the Clock ... 35
17.4 Firmware Updates ... 35
17.5 A Failed Controller ... 35

18 Recommended Values ... 36

19 Conformance ... 37
19.1 Controllers ... 37
19.2 Dosers ... 37
19.3 Configuration Documents ... 38
19.4 Schema and Tools ... 38
19.5 Test Procedures ... 38

20 Security Considerations ... 43

Appendix A — Approval Checklist ... 44

Appendix B — Examples (Informative) ... 44

Appendix C — Commercial Features (Informative) ... 45

Appendix D — Open Issues (Informative) ... 45

PRE-DRAFT

3 Scope

This Specification defines:

- the calibration of dosing heads, at one speed or at several;
- dosing programs: a daily amount split into doses across a window, on chosen weekdays or every few days, with spacing between heads, and fixed ratios between them;
- how doses are given, delayed, missed, and caught up, and how they are never given twice;
- container tracking, forecasts, warnings, and stopping when a container is empty;
- dosing adjusted from analyzer, manual, and laboratory results, within limits the owner sets, with plausibility checks and a fallback when results are stale or missing;
- pausing dosing on a condition, such as a high pH;
- the safety limits that controllers and dosing heads enforce;
- the head programs that dosers run by themselves, and how a controller works with them;
- records of doses, adjustments, calibrations, and refills in the Data Model; and
- a test procedure for every safety check.

This Specification does not define:

- how a controller evaluates a configuration, or the elements it hosts dosing in, which the Automation specification defines;
- the `doser` class, which the Device Classes specification defines, beyond the changes this Specification proposes (Appendix D);
- water changes made with dosing heads, which the Water Level and Water Change specification defines [OAP-WL Section 9.12];
- how analyzers are scheduled, maintained, and checked, which the Water Testing specification defines [OAP-WT];
- how alarms are delivered, which the Alarms and Notifications specification defines;
- which products to dose, how much, or the values to aim for;
- additives carried in top-off water, such as kalkwasser, which the Water Level and Water Change specification defines [OAP-WL Section 8.16]; or
- the mechanical and electrical design of dosing heads.

4 References

4.1 Normative References

- **[RFC 2119]** Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- **[RFC 8174]** Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, May 2017.
- **[RFC 8259]** Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, December 2017.
- **[JSON Schema]** "JSON Schema: A Media Type for Describing JSON Documents" and "JSON Schema Validation", draft 2020-12, https://json-schema.org/draft/2020-12.
- **[POSIX]** IEEE Std 1003.1-2024, "The Open Group Base Specifications Issue 8", the `TZ` environment variable, https://pubs.opengroup.org/onlinepubs/9799919799/.
- **[OAP-DM]** Open Aquarium Project, "Open Aquarium Data Model", Revision 0.4, `specs/data-model/spec.md`.
- **[OAP-DC]** Open Aquarium Project, "Open Aquarium Device Classes", Revision 0.5, `specs/device-classes/spec.md`.
- **[OAP-FS]** Open Aquarium Project, "Open Aquarium Fail-Safe Behavior", Revision 0.4, `specs/fail-safe-behavior/spec.md`.
- **[OAP-MB]** Open Aquarium Project, "Open Aquarium Module Bus", Revision 0.4, `specs/module-bus/spec.md`.
- **[OAP-AU]** Open Aquarium Project, "Open Aquarium Automation", Revision 0.3, `specs/automation/spec.md`.

PRE-DRAFT

- **[OAP-AN]** Open Aquarium Project, "Open Aquarium Alarms and Notifications", Revision 0.3, `specs/alarms-and-notifications/spec.md`.
- **[OAP-SA]** Open Aquarium Project, "Open Aquarium Security and Access", Revision 0.3, `specs/security-and-access/spec.md`.
- **[OAP-WT]** Open Aquarium Project, "Open Aquarium Water Testing", Revision 0.3, `specs/water-testing/spec.md`.

4.2 Informative References

- **[OAP-WL]** Open Aquarium Project, "Open Aquarium Water Level and Water Change", Revision 0.3, `specs/water-level-and-water-change/spec.md`.
- **[OAP-DP]** Open Aquarium Project, "Open Aquarium Device Protocol", Revision 0.3, `specs/device-protocol/spec.md`.
- **[Firmware]** Open Aquarium Project, "Breadboard controller and module firmware", `hardware/controller/firmware/`. Its controller gives a daily dose at most 30 minutes late, keeps the day of the last dose so that a restart cannot give it twice, and does not repeat a refused dose; its doser module keeps its daily volume across restarts.

5 Terms, Definitions, and Conventions

5.1 Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC 2119] [RFC 8174] when, and only when, they appear in all capitals, as shown here.

5.2 Terms and Definitions

Terms defined in the Device Classes specification [OAP-DC], such as device, endpoint, controller, and operation, in the Fail-Safe Behavior specification [OAP-FS], such as fail-safe behavior, inactive state, and unavailable input, in the Automation specification [OAP-AU], such as output, input, condition, function, interlock, mode, override, resolved state, and saved state, and in the Alarms and Notifications specification [OAP-AN], such as condition code and subject, have the same meaning here.

Head: One dosing head: an endpoint of the `doser` class [OAP-DC Section 9.7], and the output that a configuration binds to it.

Doser: A device with one or more heads.

Product: The liquid a head doses, such as one part of a multi-part supplement, a trace element solution, a carbon source, or a plant fertilizer.

Dose: One delivery of a volume by a head, started by one `dose` command or by the head's own program.

Program: What a head doses and when: the members of Table 7.

Dosing function: An Automation function that runs the programs of one or more heads on a controller (Section 7.2).

Head program: A program that a head runs by itself, written to its `program` attribute (Section 15.1).

Dosing day: A local date on which a head doses (Section 9.3).

Window: The part of a dosing day across which a head's doses are spread, from `start` to `end`.

Amount: The volume a head gives on one dosing day. The **normal amount** is the program's `amount`; the **planned amount** is the amount for a particular day, after any adjustment (Section 12).

Spacing group: Heads whose doses are kept apart in time, because their products react with each other (Section 9.6).

Leader, **follower:** A follower is a head whose doses are a fixed ratio of the doses of another head, its leader (Section 9.7).

Due: A dose is due from the moment its program says it should start.

Delayed: A dose that is due but has not started (Section 10.3).

Missed: A dose that was not given, for a reason other than a pause condition (Section 10.3).

Skipped: A dose that was not given because a pause condition held it back (Section 13.1).

Catching up: Adding the volume of missed doses to later doses of the same dosing day (Section 10.4).

Held: A head is held while a higher layer of the Automation precedence, such as an interlock, a mode, or an override, stops it [OAP-AU Section 17.3].

Result: A measured value of a water parameter used by measurement-driven dosing: an analyzer result, a manual test, or a laboratory result.

Accepted result: A result that passed the plausibility checks of Section 12.4.

Container: The bottle or tank that holds a head's product.

Calibration: Measuring what a head delivers in a known run, and setting the head's flow rate from it.

Calibration run: The run of a head that a calibration measures, started with the `calibrate` command.

Prime: A run of a head that fills its tubing with product, started with the `prime` command. Neither a prime nor a calibration run is a dose.

5.3 Notation

JSON member names, function types, condition codes, and the names of attributes, commands, and faults appear in `monospace`. Durations are in seconds, except members whose names end in `Days`. Volumes in programs are in mL, and rates in mL/min (Section 7.4). Units are Data Model unit codes [OAP-DM]. References to other Open Aquarium specifications name them, as in "Automation Section 17.3", or use their tags from Section 4, as in "[OAP-AU Section 17.3]". A section number with a third part refers to a numbered item of that section: "Section 9.7.3" means item 3 of Section 9.7. Numbers marked provisional are to be confirmed with the breadboard controller and owners' experience (Appendix D).

6 Introduction

6.1 Purpose

Dosing pumps add measured amounts of liquid to an aquarium: the parts of supplements that replace the calcium, carbonate, and magnesium corals take up, trace elements, nutrients, carbon sources, and plant fertilizers. They are among the most dangerous equipment in a home aquarium, because what they add cannot be taken out. A program that doses twice after a restart, a head that keeps running, a controller that raises its doses on a broken analyzer's reading, or two products that meet and precipitate can harm a whole system within hours.

Four widely used commercial controller ecosystems all calibrate dosing heads and split a daily amount across a window, and three of them adjust doses from measurements. Only some space heads apart, catch up missed doses, keep multi-part products in ratio, or forecast when containers run out, and each defines these in its own way. This Specification defines them once, for every controller and doser, in one format that both can run, and gives each safety check a test procedure.

6.2 Design Principles

1. **Two layers of limits.** The program limits what it plans; the head limits what it gives, whatever asks it to dose (Section 14).
2. **Never twice.** A dose that was sent is never sent again, and a dosing day is never dosed again, whatever happens (Sections 7.5 and 10.5).
3. **The day's amount is the most.** Delays and catching up move doses within a day; they never add to it (Sections 10.4 and 14.4).

PRE-DRAFT

4. **Results steer within limits.** A result changes the daily amount only within the owner's minimum, maximum, and daily step, and only once it is plausible; without results, dosing returns to normal (Section 12).
5. **Unknown means do not dose.** A pause condition or an empty sensor that is unknown stops doses (Sections 11.3 and 13.1).
6. **Reacting products never meet.** Heads of a spacing group never dose together, and followers stay in ratio with their leaders (Sections 9.6 and 9.7).
7. **The same program everywhere.** A program means the same on a controller and on a head (Section 15).
8. **Only a person adds product outside the program.** Manual doses, primes, and calibration runs are started only by a person (Section 10.6).

6.3 Overview

Figure 1 shows how the parts of this Specification work together for one head.

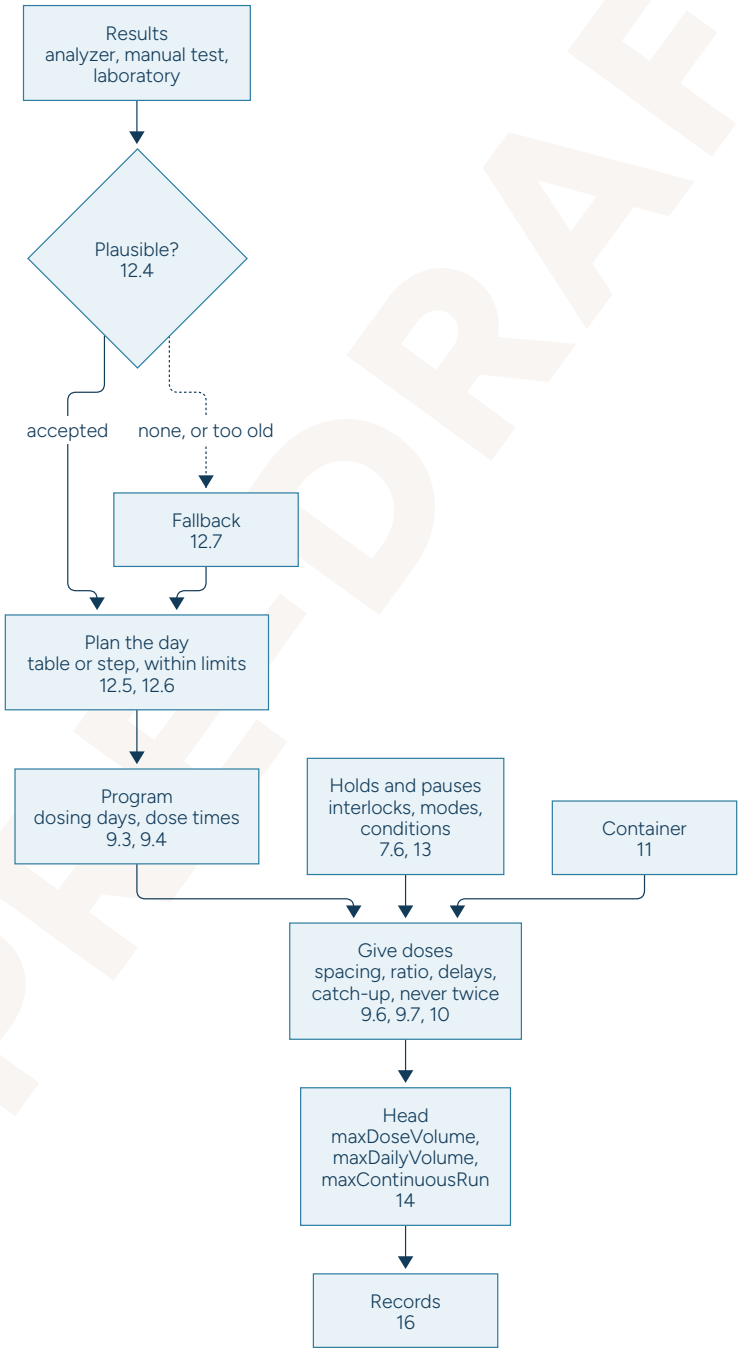

Figure 1 — The parts of dosing for one head

PRE-DRAFT

Table 1 — Parts of this Specification

Part	What it does	Section
Calibration	Measures what each head delivers, at one speed or several, and checks it at each refill.	8
Programs	Say on which days a head doses, how much, at what times, apart from which other heads, and in what ratio to another head.	9
Giving doses	Gives each dose, delays it while something prevents it, and decides when a dose is missed and whether it is caught up.	10
Containers	Track what is left, forecast when it runs out, and stop doses when a container is empty.	11
Measurement-driven dosing	Plans each day's amount from the latest plausible result, within the owner's limits.	12
Conditions	Skip doses while a condition holds, and hold heads with interlocks and modes.	13
Safety limits	The head's own limits, the day's amount, duty limits, and unexpected dosing.	14
Head programs	Let a head run its program by itself, under a controller or without one.	15

A controller runs programs as a function type that the Automation specification hosts [OAP-AU Section 11.10]. Interlocks, modes, and overrides act on heads as on any other output.

6.4 Example

A reef's alkalinity analyzer draws an air bubble with its 04:00 sample, and reports 6.1 dKH, where its last results were near 8.2 dKH. The alkalinity head plans its day from the analyzer's results with the step method, a gain of 20 mL a day per dKH, a `maxStep` of 5 mL, and a `max` of 80 mL.

1. The result is within the plausible range of 5 to 14 dKH, but it differs from the last accepted result by 2.1 dKH, more than the `maxJump` of 1.0 dKH. It is held back, and `implausible-result` is raised (Section 12.4).
2. At 08:00 the analyzer tests again, and reports 8.2 dKH. The 04:00 result is discarded, and `implausible-result` clears.
3. Had the 04:00 result been used, the step method would have wanted 44 mL a day more: 20 mL for each of the 2.2 dKH below the target of 8.3 dKH. The day's amount would still have risen by no more than `maxStep`, 5 mL, and never above `max` (Section 12.6).
4. Had the controller then sent a wrong dose, the head would have refused it: its `maxDoseVolume` is 14 mL, and its `maxDailyVolume` 120 mL (Section 14.2).
5. Had the analyzer then stopped testing, the first day planned more than 12 hours after its last result would have been planned at the normal 50 mL, and `input-unavailable` raised (Section 12.7).

The calcium head, which follows the alkalinity head in a ratio of 1 to 1, changes with it each time, so the two parts stay balanced (Section 9.7).

7 Common Requirements

7.1 Where Programs Run

1. A program runs either on a controller, which sends each dose to its head with the `dose` command, or on the head itself, which runs the head program in its `program` attribute [OAP-DC Section 9.7]. A head's `runOn` member says which (Table 8).
2. The same rules apply to both (Section 15.2).
3. A controller MUST run dosing functions in its control process, as Automation Section 17.1.3 requires, so that if they stop, the heads apply their fail-safe behavior.
4. A head that a water change drives [OAP-WL Section 9.12] is not driven by a dosing function, because an output has only one function [OAP-AU Section 11.1].

7.2 The Dosing Function

1. A dosing function is a function of the type `dosing`, which the Automation specification hosts [OAP-AU Section 11.10].

2. A dosing function has the members of Table 2, and each of its heads the members of Tables 7 and 8.

Table 2 — Members of a dosing function

Member	Type	Presence	Description
`id`, `type`, `name`, `notes`, `preset`			As for every function [OAP-AU Section 11.1].
`system`	identifier	REQUIRED	The Data Model system the function doses.
`vessel`	identifier	OPTIONAL	The vessel the doses go into. It MUST belong to `system`.
`outputs`	array	REQUIRED	The output of every head.
`heads`	array	REQUIRED	The heads, one for each output (Section 9.2).
`extensions`	object	OPTIONAL	Extensions, as in the Automation specification [OAP-AU Section 7.7].

3. `outputs` MUST list the `output` of every head, and nothing else, so that a controller that does not support dosing functions can still find their outputs, and treat them as Automation Section 19.2 requires for an unsupported function.
4. A dosing function defines its own behavior when an input is unavailable (Sections 11.3, 12.7, and 13.1), so it MUST NOT have `onUnavailable`.
5. A member described as a reference names an input or a condition of the configuration.
6. A configuration document that contains dosing functions MUST be valid against the Automation schema, and each dosing function MUST be valid against `$defs/dosing` in `schema/dosing.schema.json` [JSON Schema].
7. A configuration MAY have several dosing functions, for example one for each system. Spacing groups and ratios act only among the heads of one function.
8. Revisions 0.1 and 0.2 named the function type `org.openaquariumproject.dosing`, and kept the saved state of every dosing function under `state.extensions["org.openaquariumproject.dosing"]`, keyed by function identifier. A controller reads them as this revision's type and saved state (Section 7.7) [OAP-AU Section 7.8], and writes those of this revision.

The `doser` class has no running state; doses are actions [OAP-AU Table 5]. So a dosing function makes no demand on its heads in the Automation precedence: it starts doses while a head is not held (Section 7.6), and the layers above it stop the head with the `stop` command.

7.3 Heads

1. Each head of a dosing function is one output of the class `doser` (Section 14.6), named in the function by its output identifier.
2. In a head program, heads are named by their endpoint identifiers [OAP-DC Section 7.1], because a head knows nothing of the configuration (Section 15.1).
3. A doser with several heads presents one endpoint for each [OAP-DC Section 7.1.1], and each head has its own program, container, limits, and calibration.

7.4 Volumes, Rates, and Units

1. Volumes in programs, and in the saved state, are numbers in mL, and rates are numbers in mL/min: the units of the `doser` class [OAP-DC Section 9.7]. A program therefore means the same on a head, whose attributes each have one fixed unit, and on a controller.
2. Volumes in records, product effects, and system volumes are Data Model Quantities [OAP-DM Section 7.5].
3. A controller knows what a dose delivered from the head's `doseCompleted` event, or from its `dosedToday` (Section 10.5).
4. A controller SHOULD show volumes in the owner's preferred units.

7.5 Dosing Days and Local Time

1. The doses of a dosing day are those whose window starts on that date, even where the window ends the next day.

2. Times of day are local times in the configuration's time zone, and follow Automation Section 7.6, including its rules for local times that do not exist or occur twice.
3. A head program carries the time zone as `posixTimeZone`, because a head has no other way to know it (Section 15.1).
4. While a controller's or a head's clock is not valid [OAP-FS Section 10.5], no scheduled dose starts (Section 10.2).
5. Durations, such as `maxDelay` and `gap`, are elapsed time.
6. A controller, or a head that runs its own program, MUST NOT plan a dosing day that is not later than the last dosing day it planned, which it keeps as `day` (Table 3), or, on a head, with its record of the day's doses (Section 10.5.5). So when the clock steps back across midnight, or the time zone changes, a day that was already dosed is never dosed again.
7. When its clock steps back by more than 60 s (provisional), a controller, or a head that runs its own program, MUST treat the clock as not valid for dosing until it again shows the time it showed before the step, so that no dose falls due a second time. A controller keeps that time across a restart as the time it last saved its state [OAP-AU Section 17.8].
8. While its clock is not valid for dosing because it stepped back (item 7), a controller MUST show its owners which heads wait, those that run their own programs included, and the time they wait for. No alarm tells them otherwise: a dose that falls due meanwhile raises only `dose-missed`, an `info` alarm (Section 10.3).
9. When its clock steps back by more than 24 hours (provisional), a controller MUST ask a person to confirm the time, because a clock once set far ahead and then corrected would otherwise stop every scheduled dose until the wrong time came round again. Once a person confirms the time, the clock is valid for dosing again, and the controller plans no dosing day that is not later than the local date of the confirmed time, so that a day that may have been dosed under the wrong time is not dosed again (item 6). It then writes again the program of each head that runs its own program (Section 15.3.1). Such a head treats a program written to it after its clock stepped back by more than 24 hours as the same confirmation, and plans no dosing day that is not later than the local date on which the program was written.

7.6 Holds

1. A head is held while its resolved state [OAP-AU Section 17.3] is stop: while an interlock, an emergency or power-loss mode, an override, or a mode that lists it with `stop` holds it.
2. A head is also unable to dose while its device is lost [OAP-FS Section 7.6], while it is under local control (`controlSource local`), while it reports a fault that stops it, such as `stall`, `empty`, or `hardware-failure`, and while one of its faults is latched (Section 14.5).
3. While a head is held, a dose in progress stops, because the controller sends `stop` [OAP-AU Section 17.4], and no dose starts. What the stopped dose delivered counts as given; the rest of it is delayed, with the same due time (Section 10.3), and is given later, if at all, as a dose of its own, because a stopped dose never resumes by itself [OAP-DC Section 7.14.1].
4. Doses that fall due while a head is held, or unable to dose, are delayed (Section 10.3).
5. The Automation presets stop dosers in maintenance, water-change, emergency, and power-loss modes, and leave them to their function in feed modes [OAP-AU Table 29]. So a dose that falls due during a maintenance mode is given when the mode ends, if that is within its maximum delay; this is what Automation Table 29 leaves to this Specification.

7.7 Saved State

1. A controller MUST keep the state of each dosing function across restarts, in its saved state [OAP-AU Section 17.8], as `state.functions["<function id>"]`: an object whose `heads` member is an object keyed by output identifier.
2. Its form is defined by `$defs/savedState` in `schema/dosing.schema.json`, and Table 3 lists its members.
3. A controller MUST save it whenever it changes, and before it sends each dose (Section 10.5).
4. It is exported and imported with the configuration [OAP-AU Section 19], so that a day's doses, an adjusted amount, and container estimates continue on a new controller.
5. For a head that runs its own program, the controller saves no doses, because the head keeps them itself (Section 15.2).

Table 3 — Saved state of each head

Member	Contents
`day`	The dosing day the following members describe.
`amount`	The planned amount of that day, in mL.
`doses`	Each dose of the day, in the order they fall due: `due` (timestamp), `volume` (its volume in mL: its planned volume, plus what it catches up), `state` (`planned`, `delayed`, `dosing`, `given`, `missed`, or `skipped`), `given` (the volume it delivered), and, while it runs, `before` (the head's `dosedToday` when the dose was sent).
`adjust`	For a head with `adjust`: `amount`, the last amount planned from a result; `result`, the last accepted result, with its `value` and `time`; and `suspect`, a result held back (Section 12.4).
`container`	For a head with `container` whose head does not track it: `remaining`, the estimated volume left, and `refilled`, when it was last refilled.
`calibrated`	When the head was last calibrated.
`latched`	The codes of the head's latched faults: only `stuck-on` (Section 14.5).

7.8 Faults and Alarms

1. A controller raises the faults of dosing functions with the function as their subject [OAP-AN Section 8.2], naming the head.
2. The equipment category `doser` is hazardous [OAP-AN Table 3], so a fault that is critical for hazardous equipment, such as `stuck-on`, is critical for a head.
3. Table 4 lists the codes dosing raises. Table 5 gives the codes this Specification adds, which are listed with their default severities in `conditions.json`.

Table 4 — Codes dosing raises

Code	Defined by	When	Latches
`calibration-due`	Device Classes	A head has not been calibrated for `recalibrateDays` (Section 8.4).	No
`low`, `empty`	Device Classes	A container is low or empty (Sections 11.2 and 11.3).	No
`expired`	Device Classes	A product is past its life (Section 11.5).	No
`implausible-result`	Device Classes	A result that no testing function screens was held back or not used (Section 12.4).	No
`stuck-on`	Device Classes	A head dosed when it should not have (Section 14.5).	Yes
`input-unavailable`	Automation	A result, a pause condition, or an empty condition is unavailable (Sections 11.3, 12.7, and 13.1).	No
`command-refused`	Automation	A head refused a dose, a prime, or a calibration run (Sections 10.6 and 14.2).	No
`dose-missed`	This Specification	A dose was missed (Section 10.3).	No
`dose-adjust-limit`	This Specification	A planned amount was held at `min` or `max` (Section 12.6).	No
`calibration-drift`	This Specification	A container's real use differs from the head's count (Section 8.5).	No

Table 5 — Conditions this Specification adds

Code	Meaning	Default	Critical for	Notes
`dose-missed`	A dose of a dosing program was not given within its maximum delay (Section 10.3).	`info`	—	The cause, such as `empty`, `device-lost`, or `command-refused`, is raised as its own alarm; holds by modes and interlocks are the owner's own. With catching up, the volume is added to the day's later doses.
`dose-adjust-limit`	Measurement-driven dosing wanted a daily amount outside the owner's `min` and `max`, and the amount is held at the limit (Section 12.6).	`warning`	—	The system uses more or less than the owner expected, or the results are wrong.
`calibration-drift`	A dosing head's container emptied faster or slower than the head	`warning`	—	Every dose of the head is off by about as much. Worn tubing is the usual cause; calibrate the head.

Code	Meaning	Default	Critical for	Notes
	counted, by more than the allowed share (Section 8.5).			

7.9 Visibility

A controller MUST show the owner, for each head:

1. its next dose: when it falls due, and its volume;
2. the day's planned amount, what has been given, and the doses missed and skipped;
3. why a dose is delayed, such as "held by maintenance mode 'Maintenance', 12 min left" or "waiting 15 min after 'Alkalinity' (spacing)";
4. for a head with `adjust`, the planned amount, the result it came from, or why it is a fallback amount, the normal amount, and any limit that held it;
5. its container estimate and forecast;
6. when it was last calibrated;
7. whether it runs its own program; and
8. while its clock is not valid for dosing because it stepped back, that the head waits, and until what time (Section 7.5.8).

8 Calibration

8.1 Overview

A peristaltic head pushes product through its tubing with rollers, and a stepper-motor head counts its steps, so neither measures what it delivers. What a head delivers in a turn depends on its tubing, which wears and stretches, on the product's viscosity, on the height it pumps to, and on its speed. Heads are therefore calibrated: the owner measures what a head delivers in a known run, and the head sets its flow rate from it.

8.2 Calibrating a Head

1. A head is calibrated with the `calibrate` and `completeCalibration` commands [OAP-DC Section 9.7]: the head runs for `duration` at its default rate, or, with the `calibration-points` feature, at `rate`, the owner measures the volume, and `completeCalibration` sets `calibratedFlowRate`, or the calibration point at that rate, from it.
2. A head SHOULD be primed (`prime`) before it is calibrated, so that its tubing is full. Only a person starts a prime or a calibration run (Section 10.6.5).
3. A head SHOULD be calibrated with the product it doses, in the tubing it doses through, with the outlet at the height it doses at.
4. The calibration run SHOULD deliver at least 10 mL (provisional), measured in a graduated cylinder, or weighed and divided by the product's density. It runs no longer than the head's `maxContinuousRun`, because a head refuses a longer calibration run [OAP-DC Section 7.14.3], so a controller SHOULD propose a `duration` that delivers at least 10 mL within it, at the rate being calibrated.
5. When a calibration changes the flow rate by more than 10 % (provisional), a controller SHOULD tell the owner, and propose calibrating again before accepting it.
6. Primed and calibration volumes are not doses: they do not count towards the day's amount or `dosedToday` [OAP-DC Section 9.7], but they do come out of the container (Section 11.1).
7. A controller SHOULD record each calibration (Section 16.3), and keep its time as `calibrated` in the saved state.

8.3 Several Speeds

A head's flow is not exactly proportional to its speed, so a calibration holds only near the speed it was made at: one maker says a calibration is valid only at the exact speed it was made at, and another corrects a single calibration with factors measured at about 30 speeds.

1. A head without calibration points has one calibration, `calibratedFlowRate`, made at its `doseRate`, the rate of a dose that gives no `rate` [OAP-DC Section 9.7]. A program for such a head MUST NOT give a `rate` other than its `doseRate`, and a controller MUST NOT send it a `dose` with another `rate`.
2. A head calibrated at several speeds keeps a calibration point for each: the rate it ran at by its own reckoning, and the flow it delivered. To give a dose at a rate, it runs at the speed whose delivered flow, on straight lines between its points, is that rate. It MUST refuse a dose at a rate below its lowest point's flow or above its highest. A program MAY give such a head any `rate` within its points.
3. A head with the `calibration-points` feature keeps its points in `calibrationPoints`, each with the `rate` it ran at and the `flowRate` measured, in mL/min, and calibrates at a rate with `calibrate` and its `rate` [OAP-DC Section 9.7]. A head of Device Classes revision 0.1, which has no such feature, may present calibration points in its endpoint's `extensions` as `org.openaquariumproject.calibration-points`, the same list, and calibrates at a rate when the owner sets `doseRate` to that rate and then calibrates. A controller reads `org.openaquariumproject.calibration-points` of a head of revision 0.1 as its `calibrationPoints` [OAP-DC Table 5], and a head of revision 0.2 or later presents only `calibrationPoints`.
4. A controller MUST NOT run a program on a head that is not calibrated at the program's rate, and MUST tell the owner which rate to calibrate.

For example, a head that doses kalkwasser at 2 mL/min overnight, and a head that doses a trace element at 30 mL/min, are each calibrated at their own rate.

8.4 When to Calibrate

1. A head SHOULD be calibrated when it is installed, when its tubing is replaced, when its product changes, and every `recalibrateDays`. Makers advise calibrating monthly, or every 4 to 6 weeks; `recalibrateDays` has no default, and a controller SHOULD propose 30 (provisional). One maker counts running time instead, and advises calibrating after every 100 hours of running; a controller MAY also remind the owner by the head's `operatingHours` (Appendix D, issue 22).
2. With `recalibrateDays`, a head that has not been calibrated for that many days raises `calibration-due`, which clears when it is calibrated. Dosing continues.
3. A head MAY raise `calibration-due` itself [OAP-DC Section 9.7]; the controller then does not raise it again for that head.

8.5 Checking Calibration Against Containers

A container's real use shows whether a head still delivers what it counts.

1. When a person records a refill (Section 11.4), they MAY give the volume left in the container, measured.
2. The real use since the previous refill is the volume of that refill less the volume left. The head's count is the sum of its doses, primes, and calibration runs since that refill.
3. When more than 100 mL (provisional) has been used, and the two differ by more than 10 % (provisional) of the real use, the controller, or the head, raises `calibration-drift`, naming the head and the difference. It clears when the head is calibrated.
4. A controller SHOULD record the comparison with the refill (Section 16.3).

A difference that grows from one refill to the next usually means worn tubing.

8.6 Safety Checks

Table 6 — Calibration checks

Check	Presence	Applies to	What it requires	Section
Doses at a calibrated rate	REQUIRED	Controllers, dosers	Every dose runs at a rate at which its head is calibrated. A head calibrated only at its `doseRate` is given no other `rate`.	8.3
Calibration due	RECOMMENDED	Controllers, dosers	A head that has not been calibrated for `recalibrateDays` raises `calibration-due`, which clears when it is calibrated. Dosing continues.	8.4
Calibration check at refills	OPTIONAL	Controllers, dosers	When a person records a refill and says how much was left, the volume used is compared with the volume the head	8.5

Check	Presence	Applies to	What it requires	Section
			counted. A difference of more than the allowed share raises `calibration-drift`.	

9 Programs

9.1 Overview

A program says, for one head, how much to dose on each dosing day, at what times, apart from which other heads, and in what ratio to another head. Supplements that a system uses all day, such as alkalinity, are usually given in many small doses, and dosers offer up to 24 or more doses a day.

9.2 Members

A program has the members of Table 7, both in a head program and in a head of a dosing function. A head of a dosing function also has the members of Table 8, which stay on the controller.

Table 7 — Members of a program

Member	Type	Presence	Description
`amount`	number (mL)	REQUIRED, except with `ratio`	The normal amount: the volume to give on each dosing day.
`doses`	integer	OPTIONAL	How many doses to give on each dosing day, from 1 to 288, spread across the window. Default 1. Not with `times`.
`start`	time of day	OPTIONAL	When the window starts, and the first dose falls due. Default `00:00`. Not with `times`.
`end`	time of day	OPTIONAL	When the window ends. Default: the time of `start`, so that the window ends at the same local time on the next date. Not with `times`.
`times`	array of times of day	OPTIONAL	The times the doses fall due, instead of `doses`, `start`, and `end`.
`days`	array of days	OPTIONAL	The days of the week that are dosing days. Default: every day.
`everyDays`, `from`	integer, date	OPTIONAL	Dosing days are `from` and every `everyDays`th day after it. Both or neither.
`rate`	number (mL/min)	OPTIONAL	The rate of each dose (Section 9.5). Default: the head's `doseRate`.
`group`	string	OPTIONAL	The head's spacing group (Section 9.6): lowercase letters, digits, and single hyphens.
`gap`	number (s)	REQUIRED with `group`	The least time between a dose of this head and a dose of another head of its group.
`ratio`	object	OPTIONAL	`head` and `factor`: the head is a follower (Section 9.7).
`maxDelay`	number (s)	OPTIONAL	How late a dose may be given (Section 10.3). Default 3600 (provisional). At most 86 400.
`catchUp`	boolean	OPTIONAL	Whether missed doses are caught up (Section 10.4). Default `false` (provisional).

Table 8 — Further members of a head of a dosing function

Member	Type	Presence	Description
`output`	identifier	REQUIRED	The head's `doser` output.
`name`, `notes`	string	OPTIONAL	Shown to the owner.
`runOn`	`controller` or `head`	OPTIONAL	Where the program runs (Section 7.1). Default `controller`.
`product`	object	RECOMMENDED	The product: its `name`, `manufacturer`, and `gtin`, as a Data Model Product [OAP-DM Section 8.6], and its `effect` (Section 12.8).
`container`	object	OPTIONAL	Container tracking (Section 11): `capacity`, `lowVolume`, `lowDays`, `stopWhenEmpty`, `empty`, and `shelfLifeDays`.
`adjust`	object	OPTIONAL	Measurement-driven dosing (Section 12). Not with `ratio`.

PRE-DRAFT

Member	Type	Presence	Description
`pause`	array	OPTIONAL	References to conditions that pause the head (Section 13.1).
`recalibrateDays`	number (days)	OPTIONAL	How often the head should be calibrated (Section 8.4).
`extensions`	object	OPTIONAL	Extensions.

For example, a head that gives 50 mL of an alkalinity supplement in 12 doses a day, spaced from the other heads of its group by 30 minutes:

```
{ "output": "alk-head", "product": { "name": "Alkalinity part", "manufacturer": "Example Supplements" },
  "amount": 50, "doses": 12, "start": "00:30", "group": "balling", "gap": 1800, "catchUp": true,
  "container": { "capacity": 5000, "lowDays": 10 } }
```

9.3 Dosing Days

1. A local date is a dosing day of a head unless `days` or `everyDays` excludes it.
2. With `days`, only those days of the week are dosing days.
3. With `everyDays` N and `from`, the date `from` and every Nth date after it are dosing days, and dates before `from` are not.
4. With both, a dosing day is a date that both select.
5. A follower's dosing days are its leader's (Section 9.7).

These are the date filters of an Automation schedule [OAP-AU Section 10.6.1].

9.4 Dose Times and Volumes

1. With `times`, the doses of a dosing day fall due at those local times on that date, in order of time.
2. Otherwise, the window starts at `start` on the dosing day, and ends at the first moment after its start with the local time `end`, or, without `end`, with the local time `start`, as an Automation schedule window does [OAP-AU Section 10.6.3]. So a window without `end` ends at the same local time on the next date, or 24 hours after its start if that is sooner, as when the clocks go back. Let L be the window's length in elapsed seconds, at most 86 400, and N be `doses`. Dose k, for k from 0 to N − 1, falls due at the window's start plus $\lfloor k \times L / N \rfloor$ seconds.
3. Each dose's planned volume is the day's planned amount divided by the number of doses. Catching up adds to a dose's volume, but never changes its planned volume (Section 10.4).
4. The day's planned amount is `amount`, or, with `adjust`, the amount planned for that day (Section 12.5).
5. Each dose SHOULD be at least 0.5 mL (provisional), because smaller doses are less accurate. A controller SHOULD warn the owner about a program whose doses are smaller.

For example:

- 12 doses from `08:00` to `20:00` fall due at 08:00, 09:00, and so on until 19:00.
- 12 doses from `00:30` without `end` fall due every 2 hours, from 00:30 to 22:30.
- 2 doses from `22:00` to `06:00` fall due at 22:00 and 02:00, both on the dosing day that starts at 22:00.
- On the day daylight saving time starts in Toronto, 24 doses from `00:00` without `end` spread across 23 hours, so they fall due every 57.5 minutes.

The validator's `dose_times` function is a reference implementation of this section (Section 19.4).

9.5 Dose Rate

1. A dose runs at the program's `rate`, or, without it, at the head's `doseRate`.
2. The rate MUST be within the head's `minFlowRate` and `maxFlowRate`, and the head MUST be calibrated at it (Section 8.3).
3. A slower dose spreads the product into more water as it goes in, which matters for a product that raises the pH, such as kalkwasser, which is best added slowly.

PRE-DRAFT

9.6 Spacing Between Heads

Some products react when they meet in concentrated form. Makers of two-part supplements advise from 1 minute to an hour between the calcium and alkalinity parts, to avoid precipitating calcium carbonate, and some also advise dosing them at different places. Few dosers keep heads apart by themselves, and those that do, only within one unit or one schedule.

1. Heads of one dosing function with the same `group` form a spacing group. On a doser, heads of the device with the same `group` in their head programs form one.
2. A dose of a head MUST NOT start while a dose of another head of its group runs, nor sooner than `gap` after such a dose ended, where `gap` is the larger of the two heads' gaps.
3. A dose that item 2 holds back is delayed (Section 10.3) until it can start.
4. When doses of several heads of a group could start at the same moment, they start in the order of the heads in the function, or, on a doser, of its endpoints in its device description, each after the one before it and its gap.
5. A spacing group's heads MUST all run on the controller, or all run their own programs on one doser, because nothing else can keep them apart.
6. A controller SHOULD warn the owner when doses of two heads of a group fall due within the gap of each other, because one of them is then delayed every time, and SHOULD propose `start` times that avoid it.
7. Between products that react, a `gap` of 1800 s (provisional) is RECOMMENDED: the longest gap that more than one maker advises.
8. A controller SHOULD warn before a manual dose that would start within the gap of a dose of another head of its group (Section 10.6).

9.7 Ratio Dosing

Multi-part supplements are made to be dosed in a fixed ratio, often in equal volumes, so that the ions they add stay in balance.

1. A head with `ratio` is a follower of the head named by `ratio.head`, its leader. The leader MUST be another head of the same dosing function, or of the same doser for a head program, and MUST NOT have `ratio` itself.
2. A follower has no amount or schedule of its own, so it MUST NOT have `amount`, `doses`, `start`, `end`, `times`, `days`, `everyDays`, `from`, or `adjust`. It doses on its leader's dosing days, once for each of its leader's doses.
3. Each follower dose is `factor` times the volume its leader's dose actually delivered, and falls due when that dose ends; spacing (Section 9.6) then keeps it apart from the leader if they share a group, as they usually do.
4. When a leader's dose is missed or skipped, the follower's dose is missed or skipped with it. When a leader's dose is cut short, the follower's dose is cut in the same proportion.
5. A follower's own delays, missed doses, and catching up follow Section 10, with its own `maxDelay` and `catchUp`.
6. A follower's planned amount is `factor` times its leader's, so a leader with `adjust` changes its followers with it.
7. A follower runs where its leader runs: both on the controller, or both on the same doser.
8. When a follower cannot dose, for example because its container is empty, its leader continues, and the follower raises the cause, such as `empty` (Appendix D, issue 6).

For example, a calcium head that follows an alkalinity head with a `factor` of 1, in the same group with a gap of 1800 s, gives the same volume as each alkalinity dose, 30 minutes after it ends.

9.8 Safety Checks

Table 9 — Program checks

Check	Presence	Applies to	What it requires	Section
Dose times and volumes	REQUIRED	Controllers, dosers	Doses fall due at the times Section 9.4 gives, on the dosing days of Section 9.3, each with an equal part of the day's amount, the same way on a controller and on a head.	9.4

PRE-DRAFT

Check	Presence	Applies to	What it requires	Section
Spacing between heads	REQUIRED	Controllers, dosers	No dose of a head starts less than its group's gap after a dose of another head of its spacing group ended, or while one runs. A dose that would is delayed, and doses due together go in the order of the heads.	9.6
Ratio dosing	REQUIRED	Controllers, dosers	Each dose of a following head is `factor` times the dose its leader actually gave, and follows it. When the leader's dose is missed or skipped, so is the follower's.	9.7

10 Giving Doses

10.1 The Life of a Dose

Figure 2 shows the states of one dose, which the saved state records (Section 7.7).

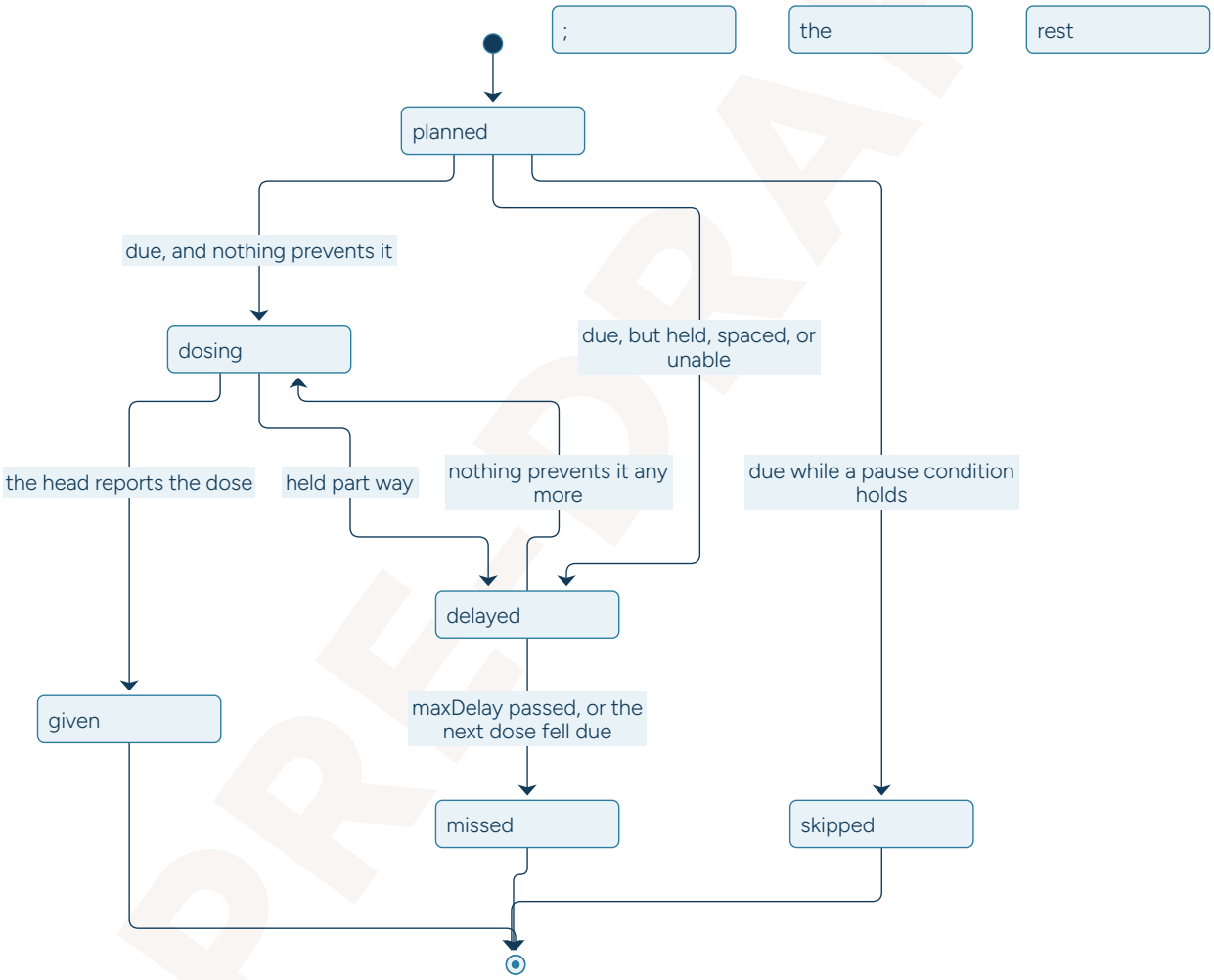

Figure 2 — The states of a dose

10.2 Giving a Dose

A dose starts when it falls due, or later while it is delayed, if all of these hold:

1. the clock is valid;
2. the head is not held, and is able to dose (Section 7.6);
3. no pause condition of the head is true or unknown (Section 13.1); a dose that falls due, or is delayed, while one is, is skipped;
4. its container is not empty (Section 11.3);

5. spacing allows it (Section 9.6);
6. for a follower, its leader's dose has ended (Section 9.7);
7. the dose would not take the day's doses above the day's planned amount (Section 14.4); and
8. no fault of the head is latched (Section 14.5).

A controller gives a dose with one `dose` command, with `volume`, and with `rate` only where the program gives one. It never sends `direction reverse` for a program's dose. It saves the dose before it sends it (Section 10.5).

A dose that the head refuses, for example because it would exceed the head's `maxDailyVolume`, raises `command-refused` [OAP-AU Section 17.4], is missed, and is not sent again [Firmware].

10.3 Delays

1. A dose that cannot start when it falls due is delayed.
2. A delayed dose starts as soon as Section 10.2 allows, if that is no later than `maxDelay` after it fell due, and before the head's next dose falls due.
3. Otherwise it is missed: it is never given, and `dose-missed` is raised, naming the head and the reason, such as "held by maintenance mode 'Maintenance'". With `catchUp`, its volume is caught up as Section 10.4 says.
4. Delays caused by spacing, and a follower's wait for its leader, count towards `maxDelay` like any other.
5. When a hold stops a dose part way, its remainder is a delayed dose with the same due time (Section 7.6.3).
6. A dose is never given on a later dosing day than its own, so `maxDelay` MUST NOT be more than 86 400 s.

`maxDelay` covers the ordinary interruptions of a day, such as a feed mode, a short maintenance mode, or a controller restart, while a dose given much later would fall on chemistry that has moved on. The breadboard controller gives its daily dose at most 30 minutes late [Firmware].

10.4 Missed Doses and Catching Up

1. Without `catchUp`, a missed dose's volume is not given.
2. With `catchUp true`, when a dose is missed, the volume it did not deliver is added, in equal parts, to the head's doses of the same dosing day that are still `planned`.
3. No dose may grow to more than 2 times (provisional) its planned volume, the volume before any catching up (Section 9.4.3), however many missed doses it catches up, or to more than the head's `maxDoseVolume`. So a dose that has already caught up a missed dose, and is then missed itself, adds to the doses after it no more than they can take within the same limit. What cannot be added is missed for good.
4. Nothing is carried to another dosing day.
5. Skipped doses (Section 13.1) are never caught up.
6. A follower catches up only its own missed doses; when its leader catches up, the follower follows the leader's larger doses.
7. After catching up, the day's doses still add up to no more than its planned amount (Section 14.4).
8. `catchUp` is off by default (Appendix D, issue 4). A controller SHOULD propose `catchUp true` for a head whose product's `effect` is on alkalinity, calcium, or magnesium, which a system uses steadily, and SHOULD NOT propose it for other products, such as carbon sources and nitrate or phosphate supplements, whose missed doses makers say never to make up.

For example, with 12 doses of 4.4 mL a day, a fifth dose missed by a maintenance mode adds 0.629 mL to each of the seven doses that remain, so the day still gives its planned amount.

Catching up keeps the daily amount the system uses, without ever doubling a dose. It is chosen for each head, because it suits some products and not others: one maker's catch-up is chosen for each head and ends at midnight, and its supplement manuals say to make up missed calcium and alkalinity doses, but never missed carbon or nitrate and phosphate doses. It is therefore off by default, and proposed by kind of product (item 8).

10.5 Never Twice

A dose given twice is an overdose, while a dose missed is the smaller harm [OAP-FS Table 3].

1. Before a controller sends a dose, it MUST save the dose with the state `dosing`, and with `before`, the head's last reported `dosedToday`.

2. A dose that was sent MUST NOT be sent again, whatever happens: a restart, a power cut, a lost `doseCompleted` event, or a lost response.
3. When the dose ends, what it delivered is the `volume` of the head's `doseCompleted` event [OAP-DC Section 9.7]. If the controller misses the event, and the head reports `dosing false`, it is the increase of `dosedToday` since `before`, on the same local date; if that cannot be known, it is taken to be the whole dose.
4. After a restart, a dose saved as `dosing` is treated as given, with what it delivered found as item 3 says. The rest of it is never given, and is not caught up, because an interrupted dose never resumes [OAP-DC Section 7.14.1].
5. A head that runs its own program MUST keep the same record across power loss: it MUST store that a dose has started before it starts it, and MUST NOT start that dose, or what was left of it, again after a restart.

The breadboard controller keeps the day of its last dose for the same reason, and its doser module stores its daily volume at once, so that a restart cannot be used to exceed `maxDailyVolume` [Firmware].

10.6 Manual Doses

1. A person may give a manual dose with the `dose` command, or prime a head with the `prime` command, with the `dose-now` operation [OAP-SA Table 6], and calibrate it with the `calibrate` operation (Section 8.2).
2. Manual doses are not part of the program: they do not count towards the day's planned amount (Section 14.4), but they do count towards the head's `maxDailyVolume`, its container, and the records, as doses of the kind `manual` (Section 16.1).
3. A controller MUST NOT send a manual dose, a prime, or a calibration run to a head that an interlock, or an emergency or power-loss mode, holds.
4. A controller SHOULD warn before a manual dose that starts within the gap of a dose of another head of its spacing group, or that is larger than the head's largest planned dose.
5. A controller MUST send `prime` and `calibrate` only when a person asks for them, never from a program, a schedule, a mode, or any other automation [OAP-DC Section 7.14.5]. So a configuration MUST NOT give a mode a `prime` or `calibrate` command for a head [OAP-AU Section 14.5]. Primes and calibration runs add product that no dose counts, so only a person who is there should start them.
6. A head refuses a prime or a calibration run that would run longer than its `maxContinuousRun`, and a prime that would take its priming time of the last 24 hours past `maxDailyPrimeTime` [OAP-DC Section 7.14]. A controller SHOULD show the person how much priming time the head has left, where it can tell, and propose a `duration` within it. A refused prime or calibration run raises `command-refused`, and is not sent again.

10.7 Safety Checks

Table 10 — Checks on giving doses

Check	Presence	Applies to	What it requires	Section
Holds stop doses	REQUIRED	Controllers	While an interlock, a mode, an override, or the leak or power-loss response holds a head, a dose in progress stops at once, and no dose starts. The rest of the dose is delayed.	10.2
Delays and missed doses	REQUIRED	Controllers, dosers	A dose that cannot start when due is given as soon as nothing prevents it, if that is within `maxDelay` of its due time and before the head's next dose falls due. Otherwise it is missed, and `dose-missed` is raised.	10.3
Catching up	OPTIONAL	Controllers, dosers	With `catchUp`, a missed dose's volume is added in equal parts to the head's doses of the same dosing day that are not yet due, each at most twice its planned volume, the volume before any catching up. Nothing is carried to another dosing day.	10.4
Never twice	REQUIRED	Controllers, dosers	A dose that was sent is never sent again, even when a restart or power cut interrupted it. The volume it delivered is counted from the head's `dosedToday`, or, if that is unknown, taken to be the whole dose.	10.5
No doses without the time	REQUIRED	Controllers, dosers	While the clock is not valid, no scheduled dose starts. When it becomes valid, each dose that fell due meanwhile is given or missed by Section 10.3.	10.2

Check	Presence	Applies to	What it requires	Section
Never a day twice	REQUIRED	Controllers, dosers	A dosing day that was planned is never planned again, even when the clock steps back across midnight or the time zone changes. A clock that steps back by more than 60 s is not valid for dosing until it catches up, or, after a step of more than 24 hours, until a person confirms the time, and the owner is shown which heads wait. A head resets `dosedToday` at most once in 23 hours.	10.5
Primes and calibration runs	REQUIRED	Controllers, dosers	A controller sends a prime or a calibration run only when a person asks, never from a program, a schedule, or a mode, and never to a head that an interlock or an emergency or power-loss mode holds. A head refuses one longer than its `maxContinuousRun`, and a prime past its `maxDailyPrimeTime`.	10.6

11 Containers

11.1 Tracking

1. A head with `container` has its container tracked: its capacity, and the volume left.
2. A head with the Device Classes `container` feature tracks it itself. The controller writes `capacity`, `lowVolume`, and `stopWhenEmpty` to the head's `containerCapacity`, `containerLowLevel`, and `stopWhenEmpty` within a minute of their differing, reads `containerRemaining`, and records refills with the head's `refill` command.
3. Otherwise the controller estimates the volume left: the volume of the last refill, less every dose, prime, and calibration run since. It keeps the estimate in the saved state (Section 7.7).
4. Primed and calibration volumes come out of the container, though they are not doses (Section 8.2.6).

11.2 Forecasts and Warnings

1. The forecast is the number of days until the container is empty: the volume left, divided by the larger of the average planned amount of the next 7 days and the average volume used each day in the last 7 days, manual doses included.
2. `low` is raised while the forecast is less than `lowDays`, or the volume left is below `lowVolume`, and clears when a refill is recorded.
3. `lowDays` defaults to 7 (provisional), which leaves time to buy more of a product, not only to refill from stock.
4. A head that tracks its own container raises `low` at its `containerLowLevel` [OAP-DC Section 9.7]; the controller does not raise `low` again for the same head, but still shows the forecast.

11.3 Empty Containers

1. `container.empty` names a condition that is true when the container is empty, such as a level switch near its bottom; a doser may present one for each container [OAP-DC Section 7.1.1].
2. While `container.empty` is true or unknown, the head starts no dose, and `empty` is raised. While it is unknown, `input-unavailable` is raised too.
3. With `stopWhenEmpty true`, an estimate at or below zero also counts as empty. The default is `false`, because an estimate drifts, and stopping wrongly misses doses the system needs, while a head whose container really is empty pumps only air. Makers differ: one doser stops when its count cannot cover the next dose, and another never stops by itself.
4. Doses that an empty container prevents are delayed, and then missed (Section 10.3).
5. After a refill that follows `empty`, a controller SHOULD propose priming the head before its next dose, because its tubing holds air.

11.4 Refills

1. A person records a refill, to the container's capacity or to a volume they give, with the `record-maintenance` operation [OAP-SA Table 6].
2. They MAY give the volume that was left, which checks the head's calibration (Section 8.5).
3. Recording a refill resets the estimate, and clears `low`, `expired`, and, unless its condition is still true, `empty`.

4. A controller SHOULD record each refill (Section 16.3).

11.5 Product Life

1. With `shelfLifeDays`, a container refilled more than that many days ago raises `expired`, which clears at the next refill.
2. Dosing continues: an expired product is usually weaker, not harmful, and stopping it could do more harm.
3. Owners SHOULD set `shelfLifeDays` from the maker's advice for products that lose strength once opened.

11.6 Safety Checks

Table 11 — Container checks

Check	Presence	Applies to	What it requires	Section
Container forecasts	RECOMMENDED	Controllers, dosers	Each head's container is estimated from every dose, prime, and calibration run, and forecast to empty from the coming days' planned amounts. `low` is raised when the forecast is less than `lowDays` or the estimate is below `lowVolume`.	11.2
Empty containers	REQUIRED	Controllers, dosers	While a container's `empty` condition is true or unknown, or its estimate is at or below zero with `stopWhenEmpty`, the head starts no dose, and `empty` is raised. Its doses are missed.	11.3
Product life	OPTIONAL	Controllers, dosers	A container refilled more than `shelfLifeDays` ago raises `expired`, which clears at the next refill. Dosing continues.	11.5

12 Measurement-Driven Dosing

12.1 Overview

A system's use of a supplement changes as its corals grow, or its plants fill in, so a fixed amount drifts away from what the system needs. A head with `adjust` plans each dosing day's amount from the latest result of one parameter, such as alkalinity, within limits the owner sets. Three of the four widely used commercial controller ecosystems adjust doses from measurements, and the fourth suggests changes for the owner to make.

12.2 Members

Table 12 — Members of `adjust`

Member	Type	Presence	Description
`input`	reference	REQUIRED	A numeric input of the parameter (Section 12.3).
`method`	`table` or `step`	REQUIRED	How the day's amount is planned (Section 12.5).
`points`	array	REQUIRED for `table`	Two to sixteen objects, each with a `value`, in the input's unit, and an `amount`, in mL.
`target`	number	REQUIRED for `step`	The value to aim for, in the input's unit.
`gain`	number (mL per unit)	REQUIRED for `step`	How much the day's amount changes for each unit the result is from the target.
`direction`	`raise` or `lower`	OPTIONAL, `step` only	Whether the product raises the parameter, as an alkalinity supplement does, or lowers it, as a carbon source lowers nitrate. Default `raise`.
`deadBand`	number	OPTIONAL, `step` only	A range around the target, in the input's unit, within which the amount does not change. Default 0.
`min`, `max`	number (mL)	REQUIRED	The least and the most the day's amount may be.
`maxStep`	number (mL)	REQUIRED	The most the day's amount may differ from the previous dosing day's.
`plausible`	object	RECOMMENDED	Plausibility limits (Section 12.4): `min`, `max`, and `maxJump`, in the input's unit.
`fallback`	`normal`, `last`, or `stop`	OPTIONAL	What the head doses when there is no usable result (Section 12.7). Default `normal` (provisional).

1. `min`, `max`, and `maxStep` have no defaults: the owner sets them (Appendix D, issue 1). A controller SHOULD propose them from the product's effect and Table 13 (Section 12.8), and sets none of them without the owner's confirmation.

2. The normal `amount` MUST be within `min` and `max`.
3. A follower MUST NOT have `adjust`; it follows its leader's planned amount (Section 9.7.6).

For example, a magnesium head planned from the owner's weekly tests:

```
{ "input": "mg-test", "method": "table",
  "points": [{ "value": 1250, "amount": 30 }, { "value": 1300, "amount": 15 }, { "value": 1400, "amount": 0 }],
  "min": 0, "max": 30, "maxStep": 10, "plausible": { "min": 900, "max": 1700, "maxJump": 150 } }
```

12.3 Results

1. `input` names an Automation input [OAP-AU Section 9]:
   - an analyzer's results, through an `endpoint` input of the class `analyzer`, which, for an analyzer that a testing function drives, reads that function's accepted results, corrected [OAP-AU Section 9.2.3];
   - the owner's tests and laboratory results, through an `observation` input, which MAY limit the methods it accepts, such as `titration` or `icp-oes`, and which passes over the observations that the Data Model says not to act on, and the results of analyzers that their testing function has not accepted [OAP-AU Sections 9.5.4 and 9.5.5]; or
   - a sensor's readings, through a `sensor` endpoint input.
2. A result's time is when its sample was taken: the analyzer result's `time`, or the observation's `time` or its sample's `collected` time [OAP-DM Section 8.4].
3. A result older than the input's `maxAge` is unavailable [OAP-AU Section 9.8]. The default for analyzer results is 48 hours [OAP-FS Section 10.2.3], and an `observation` input MUST give its own. An analyzer input's `maxAge` SHOULD be between 1.5 and 3 times the analyzer's test interval; one maker recommends 1.5 times.
4. A result with a qualifier, `<` or `>`, is not used [OAP-AU Section 9.2.2].
5. How analyzers are scheduled and corrected, and how a testing function screens their results before any input reads them, is defined by the Water Testing specification [OAP-WT]. This Specification uses results as its inputs give them, judges again only those that no testing function screens (Section 12.4), and plans no day from a result of a reagent long past its expiry (Section 12.7.7).

12.4 Plausibility

A head judges by this section the results that no testing function screens (item 5).

1. A result outside `plausible.min` and `plausible.max` is not used, and raises `implausible-result`, naming the result.
2. A result that differs from the last accepted result by more than `plausible.maxJump` is held back as suspect, and raises `implausible-result`. When the next result is within `maxJump` of the suspect one, both are accepted, because two results agree; otherwise the suspect result is discarded.
3. `implausible-result` clears when a result is accepted.
4. A result that is not used is not a new result for Section 12.5.
5. A head whose input reads results that a testing function screens [OAP-WT Section 10.3], an `endpoint` input of an analyzer that a testing function drives [OAP-AU Section 9.2.3], or an `observation` input of a parameter that a testing function of its system checks [OAP-WT Section 7.4], MUST take each result's status from the testing function: it uses the results that the testing function accepted, and MUST NOT judge them again by items 1 and 2. So a result that a retest has confirmed is used at once, rather than held back again until the next test, and `implausible-result` is raised only for results that the head judges itself. Such a head SHOULD leave `plausible` out, because its results have already passed the same checks, and a controller warns the owner about one that has it [OAP-WT Section 10.6]. Plausibility limits SHOULD be set for every head with `adjust` whose input reads results that no testing function screens. A controller SHOULD propose a plausible range well outside the owner's acceptable range for the parameter [OAP-AN Section 9].
6. A controller MAY ask an analyzer to test again, with its `test` command, when it holds a result back, as one analyzer does by itself for any result more than 0.2 dKH from the last. For an analyzer that a testing function drives, the testing function gives the retest, within its own limits [OAP-WT Section 10.4].

The validator's `screen_result` function is a reference implementation of this section (Section 19.4).

12.5 Planning the Day

1. A head with `adjust` plans each dosing day's amount once, when the day's first dose falls due, from the latest accepted result that is not older than the input's `maxAge`. A head that runs its own program is planned by the controller before the day starts (Section 15.3).
2. With `table`, the wanted amount lies on straight lines between the points, which MUST be in order of increasing `value`. Below the first point it is the first point's amount, and above the last point the last point's amount.
3. With `step`, the error is `target` less the result for `direction raise`, and the result less `target` for `lower`. The wanted amount is the last amount planned from a result, plus `gain` times the error. It is the last amount unchanged while the error is within half of `deadBand`, or when no result has been accepted since the previous dosing day was planned, so that each result is used once.
4. The wanted amount is then limited as Section 12.6 says.
5. The last amount planned from a result starts as the normal `amount`, and is kept in the saved state (Section 7.7).

The table method gives the same amount for the same result, whatever happened before; it suits results that come rarely, such as the owner's weekly tests. The step method corrects a little each day towards the target, and so finds the amount the system uses; it suits an analyzer that tests every day.

For example, with the magnesium table of Section 12.2, a result of 1290 mg/L plans 18 mL. With a step method whose target is 8.3 dKH and gain 20 mL per dKH, a result of 8.17 dKH adds 2.6 mL to the last amount.

The validator's `plan_amount` function is a reference implementation of this section and of Sections 12.6 and 12.7 (Section 19.4).

12.6 Limits

1. The planned amount MUST NOT differ from the previous dosing day's planned amount by more than `maxStep`, and MUST then be within `min` and `max`.
2. When the wanted amount was above `max` or below `min`, so that the planned amount is held at the limit, `dose-adjust-limit` is raised, naming the head and the result; it clears when a later day is planned within the limits.
3. A held amount means the system uses more or less than the owner expected, or the results are wrong; either needs a person.

`maxStep` bounds how fast results can change dosing, whatever they say. Three widely used commercial controllers bound only how far an adjusted dose may move from the normal dose, such as 35 % for alkalinity, and none of the three limits how fast it moves from day to day; `min` and `max` are the first kind of limit, and `maxStep` the second. Section 12.8 shows how to choose it from what the product does to the water.

12.7 Stale and Missing Results

1. When no accepted result is younger than the input's `maxAge` as a day is planned, the day is planned by `fallback`:
   - `normal`: the normal `amount`, as Fail-Safe Behavior Section 10.3.2 suggests;
   - `last`: the last amount planned from a result; or
   - `stop`: no doses that day.
2. While the input is unavailable, `input-unavailable` is raised [OAP-AU Section 17.9].
3. A fallback amount takes effect at once, without `maxStep`.
4. When results return, the next day is planned from them, within `maxStep` of the day before.
5. A head that runs its own program returns to its normal amount when its adjusted amount expires, whatever `fallback` says (Section 15.3).
6. A controller SHOULD propose to the owner that the last amount planned from a result become the new normal `amount` when the two have differed by more than 20 % (provisional) on 7 dosing days in a row (provisional), so that a fallback to the normal amount stays close to what the system uses (Appendix D, issue 3).

7. A controller MUST NOT plan a day's amount from a result of a test that used a reagent more than 30 days (provisional) past its expiry [OAP-WT Section 9.4.3]. When the latest accepted result is such a result, the day is planned at the normal `amount`, whatever `fallback` says, and without `maxStep`, as a fallback is (item 3), and is recorded with the basis `expired-reagent` (Section 16.2). An aged reagent biases results slowly, so its results stay recorded, shown, and used by alarms, while dosing no longer follows them.

Commercial controllers return to the normal dose when a result is bad or missing. Which fallback is safest depends on the product, and on how far the system's use has moved from the normal amount (Appendix D, issue 3).

12.8 Product Effects

1. A product's `effect` says that 1 mL of the product changes `parameter` by `value`, in `unit`, in `volume` of water, as product makers state it. `value` is negative for a product that lowers the parameter.
2. With it and the system's volume, the sum of its circulating vessels [OAP-DM Section 8.2], a controller can show the change each dose makes, and SHOULD:
   - propose a `gain` that would correct about half of an error in a day;
   - propose a `maxStep` that changes the parameter by no more than a quarter of Table 13 a day (provisional);
   - propose a `min` and a `max` no further from the normal `amount` than the volume that changes the parameter by Table 13 in a day; and
   - warn the owner when `maxStep`, or `max` less the normal `amount`, could change the parameter by more than Table 13 recommends.
3. These figures are upper bounds: the system uses some of what is dosed.

For example, a product of which 1 mL raises 100 L by 0.1 dKH raises a 500 L system by 0.02 dKH for each mL. A `maxStep` of 5 mL then changes the daily rise by no more than 0.1 dKH, and a `max` 30 mL above the normal amount adds no more than 0.6 dKH a day.

Table 13 — Recommended largest change of a parameter in a day by dosing

Parameter	Largest change in a day	Why
Alkalinity (`alkalinity`)	1.4 dKH	One supplement maker advises raising alkalinity by no more than 0.5 meq/L, 1.4 dKH, a day; others cap the daily dose instead.
Calcium (`calcium`)	20 mg/L	One supplement maker advises raising calcium by no more than 20 mg/L a day.
Magnesium (`magnesium`)	10 mg/L	One supplement maker advises raising magnesium by no more than 10 mg/L a day; others allow 50 to 100 mg/L. The lower figure is recommended until owners' experience says otherwise.

12.9 Safety Checks

Table 14 — Checks on measurement-driven dosing

Check	Presence	Applies to	What it requires	Section
Planning the day	REQUIRED	Controllers	Each dosing day's amount is planned once, when its first dose falls due, from the latest accepted result, by the table or the step method. A step method uses each result only once.	12.5
Owner's limits	REQUIRED	Controllers	A day's planned amount is never below `min` or above `max`, and never differs from the previous dosing day's by more than `maxStep`. An amount held at `min` or `max` raises `dose-adjust-limit`.	12.6
Plausible results	REQUIRED	Controllers	A result outside the plausible range is not used. A result that differs from the last accepted one by more than `maxJump` is held back until a second result confirms it. Each raises `implausible-result`. A head whose results a testing function screens takes each result's status from it, and does not judge it again.	12.4
Stale and missing results	REQUIRED	Controllers	When no accepted result is younger than the input's maximum age, the day is planned by `fallback`: the normal amount, the last amount planned from a result, or no doses. `input-unavailable` is raised. A day	12.7

Check	Presence	Applies to	What it requires	Section
			whose latest result came from a reagent more than 30 days past its expiry is planned at the normal amount.	

13 Conditions

13.1 Pausing on a Condition

1. `pause` lists references to conditions [OAP-AU Section 10], or to inputs that are true or false.
2. While any of them is true or unknown, the head's doses that fall due are skipped: they are not given, not caught up, and recorded as skipped. A dose in progress finishes.
3. The doses of the head's followers are skipped with them (Section 9.7.4).
4. While a pause condition is unknown, `input-unavailable` is raised.
5. A pause condition SHOULD be about the water's chemistry, such as a pH above which kalkwasser or an alkalinity supplement is not dosed, or a nutrient level above which more of it is not dosed. Makers' examples stop alkalinity or kalkwasser above a pH of 8.40 or 8.45.

For example, an alkalinity head that pauses while the pH is above 8.45, with a hysteresis of 0.05:

```
{ "id": "ph-high", "type": "compare", "input": "ph", "above": 8.45, "hysteresis": 0.05 }
```

with `"pause": ["ph-high"]` in the head.

A pause means the water does not need the product now. Caught up later, the doses it held back would all be added at once, which is what the pause was there to prevent.

13.2 Interlocks and Modes

1. Automation interlocks and modes hold heads (Section 7.6), and their doses are delayed, not skipped. Where a dose should wait rather than be skipped, a `block` interlock fits better than a pause.
2. Each head SHOULD require the return pump, or the pump that carries its doses into the system, with an Automation `requires` interlock [OAP-AU Section 13.3] and a `delay` of 60 s (provisional), so that doses go into moving water, and do not collect in a still sump.
3. A feed mode that stops the return pump then holds the heads through that interlock, and their doses are given after it.
4. A leak emergency mode SHOULD stop dosers, as Water Level and Water Change Table 15 recommends [OAP-WL Table 15]. That specification's leak lockout holds only the outputs of top-off, water change, and RO/DI functions [OAP-WL Section 11.2.3], so a head stops on a leak only through an emergency mode, or an interlock, that the owner has configured. The lockout does not hold dosing heads, which add little water, so that a leak does not stop dosing until a person clears a lockout [OAP-WL Appendix D, issue 25].

13.3 Safety Checks

Table 15 — Condition checks

Check	Presence	Applies to	What it requires	Section
Pausing on a condition	REQUIRED	Controllers	While a head's pause condition is true or unknown, its doses that fall due are skipped, and are never caught up; the doses of heads that follow it are skipped too.	13.1
Doses into moving water	RECOMMENDED	Configurations	Each head requires the return pump, or the pump that carries its doses away, with an Automation `requires` interlock, so that doses held while it is stopped are given once it runs again.	13.2

14 Safety Limits

14.1 Two Layers

Every head has two independent layers of limits:

1. the program's: the day's planned amount (Section 14.4), and, with `adjust`, `min`, `max`, and `maxStep` (Section 12.6); and
2. the head's own: `maxDoseVolume`, `maxDailyVolume`, `maxContinuousRun`, and, for primes, `maxDailyPrimeTime`, which the head enforces whatever asks it to dose or prime [OAP-DC Sections 7.8 and 7.14].

The controller also watches each head for doses it did not send (Section 14.5).

14.2 Limits on the Head

1. Every head that a program drives MUST have `maxDoseVolume` and `maxDailyVolume` set. A controller MUST NOT run a program on a head without them, and MUST tell the owner what to set.
2. A head MUST refuse a dose that would exceed either limit, from any source, and MUST stop a dose in progress when its `dosedToday` reaches `maxDailyVolume` [OAP-DC Section 7.8].
3. A head that runs its own program MUST have both set before it runs it.
4. A head MUST keep `dosedToday` across power loss. While it does not know when local midnight is, it MUST NOT reset `dosedToday`, which errs on the side of refusing doses [Firmware]. It MUST NOT reset `dosedToday` at a local midnight that comes less than 23 hours (provisional) of elapsed time after it last reset it, so that a clock that steps back, or a change of time zone, cannot reset it twice in one day.
5. A refused dose raises `command-refused`, and is missed (Section 10.2).
6. A head that implements `prime` MUST have `maxDailyPrimeTime` set, as Device Classes requires from its revision 0.3 [OAP-DC Section 7.14.4].
7. A head learns when local midnight is from the `posixTimeZone` that its controller writes in its `program` (Sections 15.1.1 and 15.3.2), or, over the Device Protocol, from the UTC offset of the time its controller sends it [OAP-DP Section 12.3.2]; otherwise it uses its own setting [OAP-MB Section 9.4.9]. A head on the module bus without the `local-program` feature has only its own setting, so a controller SHOULD tell the owner, when it binds such a head, to set the head's time zone, because a head that does not know local midnight never resets `dosedToday`, and refuses doses once it reaches `maxDailyVolume`.

14.3 Choosing Head Limits

1. `maxDoseVolume` SHOULD be at least the head's largest planned dose, times the catch-up limit of 2 if it has `catchUp`, and no larger than that needs.
2. `maxDailyVolume` SHOULD be about 1.5 times (provisional) the largest amount the program may plan for the head: `max` for a head with `adjust`, `amount` for other heads, and `factor` times the leader's for a follower, plus the manual doses the owner expects.
3. A controller SHOULD propose these values, and SHOULD warn the owner when a head's limits are below what its program needs, so that doses would be refused, or more than twice what this section proposes.

For example, an alkalinity head whose `max` is 80 mL, in 12 doses with `catchUp`, has a largest planned dose of 6.7 mL, so a `maxDoseVolume` of 14 mL and a `maxDailyVolume` of 120 mL.

14.4 The Day's Amount

1. The doses a program gives a head on a dosing day, caught-up doses included, MUST NOT add up to more than the day's planned amount.
2. Before each dose, the controller, or the head running its own program, checks this, and reduces a dose that would exceed it to what is left.
3. When a head's program changes during a dosing day, its remaining doses are planned again from the new program, and the day's doses, given and to come, still MUST NOT add up to more than the new day's planned amount.
4. Manual doses, primes, and calibration runs are not counted (Sections 8.2 and 10.6).

PRE-DRAFT

14.5 Unexpected Dosing

1. A controller MUST watch every head that a dosing function drives. A head that does not report `controlSource local` doses unexpectedly when:
   - it reports `dosing` for more than 5 s (provisional) while no dose, prime, or calibration run that the controller sent is running, where a prime or a calibration run counts as running for its `duration` from when the head accepted it; or
   - its `dosedToday` rises, within a local day, by more than the doses the controller sent, and those that the head reported while it was under local control, plus 1 mL and 5 % of what it sent (provisional). A person who runs a head at its own controls doses on purpose, and a controller sends a head under local control no command but the stops that protect the system [OAP-AU Section 17.4.2].
2. The controller then sends `stop`, and raises `stuck-on` [OAP-DC Section 8.4], naming the head. `stuck-on` latches: the head starts no dose until a person clears it, with the `clear-latch` operation [OAP-SA Table 6], and the latch is kept in the saved state across restarts.
3. For a head that runs its own program, the controller does not know when each dose falls due, so it compares the head with its program's days instead. Such a head doses unexpectedly when its `dosedToday` rises, within a local day, by more than the planned amounts of the dosing days whose windows overlap that date, adjusted amounts included (Section 15.3.7), the manual doses the controller sent, and those that the head reported while it was under local control, plus 1 mL and 5 % (provisional) of that total; or when it reports `dosing` for more than 5 s (provisional) while a hold without `skip` holds it (Section 15.3.3), other than for a manual dose, a prime, or a calibration run that the controller sent.

A head can dose unexpectedly through a firmware fault, a motor driver that fails on, another controller or app that sends doses, or a pump whose switch fails closed.

14.6 Dosing Heads Only

1. A dosing function drives only outputs of the class `doser` (Appendix D, issue 5).
2. A pump on a switched outlet doses for as long as the outlet is on, and has no dose or daily limit of its own; only the controller would limit it, and a switch that fails closed would dose until its container is empty.
3. Such a pump SHOULD be driven through a module that presents it as a `doser` endpoint, and enforces the limits of Section 14.2 itself, as the breadboard doser module does [Firmware].

14.7 Duty Limits

1. A head's `maxContinuousRun` limits how long it runs without a rest: the head refuses a dose, a prime, or a calibration run that would run longer [OAP-DC Section 7.14.3]. Every head of Device Classes revision 0.5 or later has it [OAP-DC Section 9.7]; a head of an earlier revision may lack it, and is then limited only by its other limits.
2. A controller MUST NOT send a dose whose running time, its volume divided by its rate, is longer than the head's `maxContinuousRun`. It gives such a dose in parts, each no longer than `maxContinuousRun`, with a rest after each at least as long as the part (provisional). The parts are one dose for Sections 10 and 14.4.
3. A head that runs its own program does the same.
4. A controller MUST NOT send a prime or a calibration run longer than the head's `maxContinuousRun` (Section 10.6.6).

A dose long enough to need this usually means too few doses, or too slow a rate. One maker refuses doses longer than 8 minutes; another says to run its heads no more than 15 minutes at top speed, then rest as long.

14.8 Siphoning and Placement

This section is informative.

Siphoning is the main documented way a stopped doser overdoses: a container above a dosing outlet can siphon through its head, and makers say containers must sit below the doser, and outlets must never be submerged. Containers SHOULD sit below the dosing outlets, and dosing lines SHOULD end above the water's surface, or have check valves, so that neither the container nor the system can siphon through them. Few

dosers ship with check valves. Dosing outlets SHOULD be in strong flow, apart from each other for products that react, as two makers advise, and away from probes, whose readings a nearby dose would distort.

14.9 Safety Checks

Table 16 — Safety limit checks

Check	Presence	Applies to	What it requires	Section
Limits on the head	REQUIRED	Controllers, dosers	Every head that a program drives has `maxDoseVolume` and `maxDailyVolume` set, and enforces them whatever it is asked. A controller does not run a program on a head without them, and tells the owner.	14.2
The day's amount	REQUIRED	Controllers, dosers	The doses a program gives a head on a dosing day, caught-up doses included, never add up to more than the day's planned amount.	14.4
Unexpected dosing	REQUIRED	Controllers	A head that is not under local control and reports dosing when it should not, or whose `dosedToday` rises by more than its doses explain, is stopped and held, and `stuck-on` is raised and stays until a person clears it. A head that runs its own program is compared with its program's days.	14.5
Dosing heads only	REQUIRED	Controllers, configurations	Programs drive only `doser` outputs, whose heads enforce their own dose and daily limits. A pump or a switched outlet is not a head.	14.6
Duty limits	REQUIRED	Controllers	No part of a dose runs longer than the head's `maxContinuousRun`. A longer dose is given in parts, each followed by a rest at least as long. No prime or calibration run is sent longer than `maxContinuousRun`.	14.7

15 Heads That Run Their Own Programs

15.1 Head Programs

1. A head with the `local-program` feature runs the head program in its `program` attribute [OAP-DC Section 9.7], written as JSON text [RFC 8259] without whitespace between tokens. An empty `program`, or one whose only members are `posixTimeZone` and `extensions`, means that the head follows `dose` commands; the second also gives the head the local time zone (Section 15.3.2).
2. A head program has the members of Table 7, and those of Table 17, and MUST be valid against `$defs/headProgram` in `schema/dosing.schema.json`.
3. In a head program, `ratio.head` is the endpoint identifier of another head of the same doser.
4. A head MUST reject a write of a program it cannot run in full, and keep its previous program [OAP-DC Section 7.13.2].
5. Members whose values are their defaults SHOULD be left out.
6. A head that runs its own program MUST implement POSIX `TZ` strings with rules, in the `M`, `J`, and plain-number forms, and with transition times [POSIX], so that it follows any `posixTimeZone` that a controller writes (Section 15.3.1).

Table 17 — Further members of a head program

Member	Type	Presence	Description
`posixTimeZone`	string	OPTIONAL	The local time zone, as a POSIX `TZ` string [POSIX], such as `AEST-10`, as in Automation. A controller always writes it (Section 15.3.1). Without it, the head uses its own setting.
`adjusted`	object	OPTIONAL	An adjusted amount, written by a controller (Section 15.3): `amount`, in mL, and `until`, a timestamp.
`extensions`	object	OPTIONAL	Extensions.

For example, the `program` of a calcium head that follows the alkalinity head `head_1` of the same doser:

```
{"ratio":{"head":"head_1","factor":1},"group":"two-part","gap":1800,"catchUp":true,"posixTimeZone":"CET-1CEST,M3.5.0,M10.5.0/3"}
```

15.2 Running a Head Program

1. A head with a program follows Sections 9, 10, 11, and 14 by itself, with its own clock: its dosing days and dose times, spacing with the heads of its doser, ratios, delays, catching up, never dosing twice, its container, its limits, and its duty limits.
2. It doses whether or not a controller is connected: while its controller controls it, and, with `failSafeAction continue`, while it has lost contact [OAP-FS Section 7.3]. With `failSafeAction state`, it stops dosing when it loses contact, as Fail-Safe Behavior requires.
3. With `failSafeAction continue`, it keeps everything its controller last commanded when it loses contact, including `enabled` and any pending revert [OAP-DC Section 8.3.2]: it doses only while it is enabled. So a hold that its controller sent lasts until its `revertAfter` falls due, or, for a hold sent without one, until a person ends it, through the controller once it returns, or with the head's own controls for as long as that local control lasts [OAP-FS Section 12.4].
4. It reports each dose with a `doseCompleted` event.
5. It MUST keep its record of the day's doses (Section 10.5), and its `enabled`, across power loss [OAP-DC Section 8.3.4], so that a head that was held when it lost power is still held after power-up.
6. After power-up, a head whose `powerOnBehavior` is `fail-safe` and whose `failSafeAction` is `continue` runs its program again, unless a hold that survived the power cut stops it [OAP-FS Section 9.1]. A dose that the power cut interrupted does not resume (Section 10.5.5).
7. Without a valid clock, it gives no scheduled dose.
8. A head cannot read Automation inputs, so `pause`, `adjust`, and `container.empty` act through the controller (Section 15.3).
9. A head keeps a hold that its controller latched (Section 15.3.4) through loss of contact and power cuts, whatever its `failSafeAction` and `powerOnBehavior`, until a controller ends it [OAP-FS Section 8.4].

15.3 Under a Controller

1. For a head whose `runOn` is `head`, the controller writes its head program within a minute of its differing from the head's `program`: the members of Table 7 that the head has, with `ratio.head` as the leader's endpoint identifier, the time zone as `posixTimeZone`, and any `adjusted` amount. It also writes the container settings (Section 11.1). The time zone is the configuration's `posixTimeZone`, or, for a configuration without one, a POSIX `TZ` string that the controller derives from the configuration's `timeZone`, and that gives the same local time for every date from now to at least one year later, as Automation Section 7.2 requires of `posixTimeZone` [OAP-AU Section 7.2]. A controller that derives it writes the program again before that year ends, so that the head never falls back on a time zone of its own.
2. For a head whose `runOn` is `controller`, the controller writes, if the head has the `local-program` feature, a `program` whose only member is `posixTimeZone`, the time zone of item 1, and writes it again as item 1 says. The head then follows `dose` commands, and knows when local midnight is, at which it resets `dosedToday` (Section 14.2.4), so that `dosedToday` counts one day at a time, and the head does not refuse doses once the doses of several days add up to `maxDailyVolume`. A controller does the same for a head that a water change drives [OAP-WL Section 9.12].
3. A controller holds a head that runs its own program with `set` [OAP-DC Section 9.7], whose parameters are `enabled` (boolean), `skip` (boolean, OPTIONAL, default `false`), and `revertAfter` (number, s, OPTIONAL):
   - with `enabled false`, the head starts no dose, and the doses that fall due meanwhile are delayed (Section 10.3); it also stops a dose in progress, unless `skip` is `true`;
   - with `skip true`, those doses are skipped instead, as for a pause condition (Section 13.1);
   - with `enabled true`, or when `revertAfter` falls due, it resumes its program, and a dose that the hold stopped does not resume [OAP-DC Section 7.14.1]; and
   - it reports whether it is held as `enabled`. A head of Device Classes revision 0.1 has no `set`. A controller holds one that implements the provisional extension command `org.openaquariumproject.set-program`, which has the same parameters, and reads its `org.openaquariumproject.program-enabled`, in its endpoint's `extensions`, as `enabled` [OAP-DC Table 5].
4. The controller sends the hold whenever a layer of the Automation precedence holds the head (Section 7.6), and while a pause condition is true or unknown, with `skip true`, or while `container.empty` is true or unknown. A hold whose end is known is sent with `revertAfter` [OAP-AU Section 17.5.1], so that it ends at

its revert even if the controller fails: such as a feed mode's, an override's, or the stop of a `requires` interlock while the pump it requires is paused, which is sent with `revertAfter` for the pause's end plus the interlock's `delay` [OAP-AU Section 17.5.5]. A hold with no known end that an interlock or an emergency mode demands is latched [OAP-AU Section 17.5.4; OAP-DC Section 7.4.13], so the head keeps it through loss of contact and power cuts, in place of its fail-safe behavior, until a controller ends it [OAP-FS Section 8.4]. Other holds with no known end, such as a pause condition's, an empty container's, or a power-loss mode's, are sent without `revertAfter`, and are not latched, so a head with `failSafeAction continue` whose controller fails meanwhile stays held until a person ends the hold (Section 15.2.3). The hold and the head's `stop` may go in either order, because a command that ends a dose leaves the hold's revert as it is [OAP-DC Section 7.4.7].

5. A controller MUST NOT let a head run its own program unless the head implements `set` with `enabled`, or, for a head of Device Classes revision 0.1, `org.openaquariumproject.set-program`; otherwise it runs the program itself.
6. For a head with `adjust`, the controller plans each dosing day as Section 12.5 says, 600 s (provisional) before the day's first dose falls due, and writes the planned amount as `adjusted`, with `until` 3600 s (provisional) after that dose falls due. For `fallback stop`, it writes an amount of 0.
7. A head uses `adjusted.amount` as the planned amount of a dosing day whose first dose falls due before `adjusted.until`, and its `amount` otherwise. So if the controller fails, the head doses its normal amount from the next day.
8. The controller watches the head for unexpected dosing against its program (Section 14.5.3), and records its doses from its `doseCompleted` events (Section 16.1).

15.4 Size

1. On the module bus, a value holds at most 256 bytes [OAP-MB Section 11.6]. A head program for a module MUST fit in 256 bytes as Section 15.1 encodes it.
2. The Device Protocol carries messages to devices of at least 8192 bytes [OAP-DP Section 9.3], so a head program over the network is not limited this way.
3. A head program with a window, spacing, a maximum delay, catching up, a time zone, and an adjusted amount takes about 220 bytes, so the limit leaves little room for `times` lists or extensions.

15.5 Safety Checks

Table 18 — Checks on heads that run their own programs

Check	Presence	Applies to	What it requires	Section
The same rules on a head	REQUIRED	Dosers	A head with a program follows Sections 9 to 11 and 14 by itself, with its own clock, whether or not a controller is connected.	15.2
Holding a head's program	REQUIRED	Controllers, dosers	A controller holds a head that runs its own program with `set` and `enabled`, with `revertAfter` for a hold that ends, and latches a hold with no known end that an interlock or an emergency mode demands. A held head stops any dose and starts none, and its doses are delayed as on a controller. A head keeps a latched hold, and, with `failSafeAction continue`, any other hold, when it loses its controller, and across a power cut.	15.3
Adjusted amounts expire	REQUIRED	Controllers, dosers	A head uses an adjusted amount only for dosing days whose first dose falls due before its `until`. A controller writes it before the day starts. Without a newer one, the head returns to its normal amount.	15.3
Writing head programs	REQUIRED	Controllers	A controller writes each head program to its head, as Section 15.1 encodes it, within a minute of their differing, and writes the head's container settings. It writes a program with only the time zone to a head with the `local-program` feature that it runs itself.	15.3

16 History

16.1 Dose Records

1. A controller SHOULD record each dose of a program, whether given, missed, or skipped, and each manual dose, as a Data Model event of the type `dose` [OAP-DM Section 8.6]:
   - `time`, when it started, or when it fell due if it was not given; `end`, when it ended;
   - `system`, and `vessel`, the function's vessel;
   - `equipment` and `channel`, the head;
   - `amount`, the volume it delivered, in a volume unit, and 0 if it was missed or skipped; and
   - `product`, the head's product as a Data Model Product: its `name`, `manufacturer`, and `gtin`, without its `effect`, which a Data Model Product does not have.
2. Its `extensions["org.openaquariumproject.dosing"]` holds `$defs/doseRecord` of `schema/dosing.schema.json`: `kind` (`scheduled` or `manual`); `planned`, the dose's planned volume, before catching up (Section 9.4.3); `caughtUp`, the volume it caught up (Section 10.4); `function`; `rate`; `result` (`given`, `interrupted`, `missed`, or `skipped`); and `reason`.
3. A controller MAY instead record a head's given doses of one dosing day as one event, when it records many doses: `occurrences` the number of doses, `time` the start of its first, `end` the end of its last, and `amount` their total [OAP-DM Section 8.6.2]. It MUST then still record missed and skipped doses one by one, and manual doses are not part of the total. A dosing day whose window crosses midnight spans two local dates: a controller that writes a Data Model document of revision 0.3 or later records it as one total, whatever dates it spans, as that revision allows [OAP-DM Section 8.6.2], and one that writes a document of revision 0.2, which totals the occurrences of one local date, records it as one total for each date.
4. For a head that runs its own program, the controller records the doses its `doseCompleted` events report.

16.2 Adjustments

1. A controller SHOULD record each dosing day's planned amount of a head with `adjust` as an event of the type `dose-adjusted` [OAP-DM Section 8.6], with `time` when it was planned, `equipment` and `channel` the head, and `extensions["org.openaquariumproject.dosing"]` holding `$defs/adjustmentRecord`: `day`, `amount`, `normal`, `basis`, the `result` used, and `limitedBy`, the limit that held it, if any. `basis` says how the amount was planned: `result`, from a result; `no-new-result`, unchanged by the step method for want of a new result (Section 12.5.3); `fallback-normal`, `fallback-last`, or `fallback-stop`, by its `fallback` (Section 12.7.1); or `expired-reagent`, at the normal amount because the latest result came from a reagent long past its expiry (Section 12.7.7).
2. The owner can then see why each day's amount changed, and check the method against later tests.

16.3 Calibrations and Refills

1. A controller SHOULD record each calibration of a head as a Data Model event of the type `calibration`, with `equipment` and `channel` the head, and `calibration` a Calibration Record [OAP-DM Section 8.12]:
   - `outcome`, and `parameter` and `model` both `flow-rate`;
   - `points`, with one point for the run: its `reference`, the flow measured, and its `reading`, the rate the head ran at by its own reckoning, both in mL/min;
   - `result`, whose `rate` is the flow rate the calibration set, and `previous`, whose `rate` is the one it replaced. Its `extensions["org.openaquariumproject.dosing"]` holds `$defs/calibrationRecord`: `duration`, how long the head ran, and `measured`, the volume measured.
2. A controller SHOULD record each refill of a head's container as an event of the type `refill` [OAP-DM Section 8.6], with `equipment` and `channel` the head, `amount` the volume added, `product` the head's product, as in Section 16.1, and `extensions["org.openaquariumproject.dosing"]` holding `$defs/refillRecord`: `left`, `estimated`, and `expires`.
3. A controller reading a Data Model document of revision 0.1 reads the provisional event types of Dosing revision 0.1, `org.openaquariumproject.dose-adjusted` and `org.openaquariumproject.container-refill`, as `dose-adjusted` and `refill`, and the members `rate`, `flowRate`, and `previous` of an older `calibrationRecord` as the reading, the result, and the previous result of a Calibration Record.

17 Fail-Safe Behavior

17.1 Outputs

Table 19 gives the RECOMMENDED fail-safe configuration [OAP-FS Section 13] of heads. It follows the Fail-Safe Behavior recommendation for the category `doser` [OAP-FS Table 3].

Table 19 — Recommended fail-safe configuration of heads

Head	On loss of contact	After power-up	On battery	Notes
Run by a controller	`stop`	`fail-safe`	`stop`	A missed dose is safer than an overdose. A dose in progress stops, and does not resume.
Running its own program	`continue`, or `stop`	`fail-safe`	`stop`	With `continue`, it keeps everything its controller last commanded: it doses its program while it is enabled, with its normal amount once an adjusted amount expires, and without pause conditions, and a hold lasts until its `revertAfter` falls due, or until a person ends it. After power-up it runs its program again, unless a hold that survived the power cut stops it, and a dose that the power cut interrupted does not resume. With `stop`, it doses only under its controller. With either, a hold that an interlock or an emergency mode latched lasts until a controller ends it (Section 15.3.4).

17.2 Lost Inputs

Dosing defines what it does when an input is unavailable, as Fail-Safe Behavior Section 10.3 requires.

Table 20 — What dosing does when an input is unavailable

Input	Behavior	Section
`adjust.input`	The day is planned by `fallback`; by default, the normal amount.	12.7
A pause condition	The head's doses are skipped.	13.1
`container.empty`	The head starts no dose; its doses are missed.	11.3
The head's device	Doses are delayed, then missed.	7.6
The clock	No scheduled dose starts.	10.2

17.3 Restarts and the Clock

1. After a restart, a controller restores its saved state (Section 7.7), and follows Fail-Safe Behavior Section 9.4.
2. A dose saved as `dosing` is treated as given, and is never sent again (Section 10.5). A dosing day that was already planned is never planned again, even if the clock has stepped back (Section 7.5).
3. Doses that fell due while the controller was off are delayed or missed as Section 10.3 says, measured from their due times, once the clock is valid.
4. While the clock is not valid, no scheduled dose starts.
5. Latched faults stay latched.

17.4 Firmware Updates

1. A doser does not start a firmware update while one of its heads doses [OAP-FS Section 11.2].
2. A controller SHOULD NOT start its own firmware update, or a restart that it has scheduled, while a dose runs.
3. Doses that fall due during an update are delayed.

17.5 A Failed Controller

When the controller fails:

1. heads that it runs stop at their fail-safe timeout, and give no doses until it returns;
2. heads that run their own programs, with `failSafeAction continue`, keep everything the controller last commanded [OAP-FS Section 7.3]: a head that was not held keeps dosing, with its normal amount once its adjusted amount expires, and without its pause conditions; a head held with `revertAfter`, such as for a feed mode, or by an interlock that requires a return pump that a feed mode pauses, resumes its program

when the revert falls due; and a head held without one, such as by a pause condition or an empty container, stays held until a person ends the hold;

3. a head whose hold an interlock or an emergency mode latched stays held, whatever its fail-safe configuration, through power cuts too, until a controller ends the hold [OAP-FS Section 8.4]; and
4. the controller cannot raise an alarm, so the alarm outputs and heartbeats of the Alarms and Notifications specification tell the owner [OAP-AN Section 18].

Owners SHOULD run heads whose products need a pause condition, such as kalkwasser, on the controller, so that they stop with it.

18 Recommended Values

Table 21 gives the RECOMMENDED values that do not depend on the system. They are provisional, and are published in machine-readable form in `checks.json`, with the daily changes of Table 13. Values that depend on the system and its products, such as amounts, `min`, `max`, `maxStep`, and the heads' limits, are set by the owner, with the help of Sections 12.8 and 14.3.

Table 21 — Recommended values

Area	Value	Recommended	Why
Calibration	Calibration volume	10 mL	The least volume measured in a calibration. With a 10 mL graduated cylinder or a scale reading to 0.1 g, it keeps the reading error near 1 %.
Calibration	`recalibrateDays`	30 days	Makers advise calibrating monthly, or every 4 to 6 weeks, or after 100 hours of running, and after changing tubing, product, or plumbing.
Calibration	Calibration change	10 %	A calibration that changes the flow rate by more than this is shown to the owner, who may calibrate again before accepting it (Section 8.2).
Calibration	Calibration check volume	100 mL	The least real use over which a refill checks a head's count (Section 8.5); less is within the error of reading a container.
Calibration	Calibration drift	10 %	A head whose real use differs from its count by more than this needs calibrating; less is within the error of reading a container's level.
Programs	Smallest dose	0.5 mL	Heads give doses from 0.1 to 1 mL, but one maker claims ±0.05 mL a dose, a tenth of a 0.5 mL dose. Fewer, larger doses are better than many tiny ones.
Programs	`gap`	1800 s	Between products that react with each other, such as the two parts of a calcium and alkalinity supplement. Makers advise from 1 minute to 30 minutes, or an hour, to avoid precipitating calcium carbonate; spacing costs little, so the longer gap is recommended.
Giving doses	`maxDelay`	3600 s	Covers a feed mode and a short maintenance mode. A later dose falls on chemistry that has already moved on.
Giving doses	Catch-up limit	2 times	The largest a caught-up dose may grow, as a multiple of its planned volume (Section 10.4).
Giving doses	Clock step	60 s	A clock that steps back by more than this is not valid for dosing until it catches up (Section 7.5.7); smaller steps are a clock's ordinary corrections.
Giving doses	Clock step to confirm	24 h	A clock that steps back by more than this waits for a person to confirm the time, rather than for the time before the step to come round again (Section 7.5.9); smaller steps wait no more than a day.
Containers	`lowDays`	7 days	Leaves time to buy more of a product, not only to refill from stock. One maker warns 3 days ahead by default.
Measurement-driven dosing	Adjusted amount written	600 s	How long before a dosing day's first dose a controller writes the day's adjusted amount to a head that runs its own program (Section 15.3).

PRE-DRAFT

Area	Value	Recommended	Why
Measurement-driven dosing	Adjusted amount until	3600 s	How long after the day's first dose falls due the adjusted amount expires, so that it covers that day and no other (Section 15.3).
Measurement-driven dosing	Normal amount drift	20 %	When the amounts planned from results have differed from the normal amount by more than this on 7 dosing days in a row, a controller proposes a new normal amount (Section 12.7.6), so that a fallback stays close to what the system uses.
Measurement-driven dosing	Normal amount drift days	7 days	The dosing days in a row after which a controller proposes a new normal amount (Section 12.7.6): long enough to ignore a few unusual days.
Conditions	Return pump delay	60 s	The `delay` of a head's `requires` interlock on the return pump (Section 13.2): doses wait until the water is moving again.
Safety limits	Daily limit margin	1.5 times	A head's `maxDailyVolume`, as a multiple of the largest amount the program may plan for it (Section 14.3).
Safety limits	Unexpected dosing time	5 s	How long a head may report dosing with no dose sent before the controller stops it (Section 14.5).
Safety limits	Unexpected volume	1 mL	With the unexpected share, how far `dosedToday` may exceed the doses sent (Section 14.5).
Safety limits	Unexpected share	5 %	Of the volume sent: with the unexpected volume, how far `dosedToday` may exceed the doses sent (Section 14.5).
Safety limits	Duty rest	1 times	The rest after each part of a long dose, as a multiple of the part's running time (Section 14.7). One maker says to rest its heads as long as they ran; another needs three times a dose's running time between doses.
Safety limits	dosedToday reset	23 h	The least elapsed time between two resets of a head's `dosedToday` (Section 14.2.4): a day is 23 hours long when the clocks go forward.

19 Conformance

19.1 Controllers

A conforming controller:

1. runs dosing functions as Sections 7 to 14 define, including dosing days (Section 7.5), holds (Section 7.6), never dosing twice (Section 10.5), primes and calibration runs only at a person's request (Section 10.6), the day's amount (Section 14.4), and unexpected dosing (Section 14.5);
2. works with heads that run their own programs as Section 15.3 defines;
3. raises the codes of Table 4, and shows what Section 7.9 requires;
4. meets Section 17; and
5. passes the test procedures of Section 19.5 that apply to controllers.

19.2 Dosers

A conforming doser:

1. implements the Device Classes `doser` class and the Fail-Safe Behavior specification [OAP-DC Section 13.1];
2. enforces the limits of Sections 14.2 and 14.7, and those of primes and calibration runs [OAP-DC Section 7.14];
3. if it has the `local-program` feature, runs head programs as Sections 15.1 and 15.2 define, and implements `set` with `enabled` as Section 15.3 uses it; and
4. passes the test procedures of Section 19.5 that apply to dosers, with no controller connected and again under a controller's control.

PRE-DRAFT

19.3 Configuration Documents

A conforming configuration document is a conforming Automation configuration document [OAP-AU Section 20.1] whose dosing functions are valid against their schema, and meet every requirement of Sections 7 to 15 that applies to documents, including those a schema cannot express, such as references, ratios, spacing groups, and limits.

19.4 Schema and Tools

`schema/dosing.schema.json` is normative for the structure of dosing functions, head programs, the saved state, and the dosing members of Data Model events. Where it and this document differ, this document governs.

`checks.json` is normative for the safety checks, the test procedures, the recommended values, and the daily changes, and Tables 6, 9, 10, 11, 13, 14, 15, 16, 18, 21, and 22 are generated from it by `tools/render_spec_tables.py`. That tool also checks that every safety check has a test procedure, and that every MUST and MUST NOT of Sections 7 to 20 is checked by a test. Table 5 is generated from `conditions.json`.

The validator, `tools/validate_dosing.py`, checks configuration documents against the Automation validator, the schema, and the rules a schema cannot express, the head programs of device descriptions, and the dosing records of Data Model documents. It warns about choices this Specification advises against. Its `dose_times`, `screen_result`, and `plan_amount` functions are reference implementations of Sections 9.3 and 9.4, 12.4, and 12.5 to 12.7. It is informative.

The `examples` directory contains complete documents (Appendix B).

19.5 Test Procedures

Tests use a controller with simulated dosers whose `dosing`, `dosedToday`, `doseCompleted` events, and refusals the tester can watch and set, whose power the tester can remove, and whose clock the tester can set or stop, and a doser with the `local-program` feature that can run with no controller. “Within 1 s” allows for the controller’s evaluation interval of 1 s [OAP-AU Section 17.1]. Every safety check of Tables 6, 9, 10, 11, 14, 15, 16, and 18 has a test in Table 22, which follows the checks with tests of the other requirements. Each test names the requirements it checks, as sections or as numbered items of sections, and every MUST and MUST NOT of Sections 7 to 20 is checked by at least one test. A pass criterion that checks what a SHOULD asks applies to a product that does it.

Table 22 — Test procedures

Test	Applies to	Requirements	Method	Pass criteria
Doses at a calibrated rate (8.3)	Controllers, dosers	8.3.1, 8.3.2, 8.3.3, 8.3.4, 9.5.2	Give a program a `rate` other than the `doseRate` of a head calibrated at one speed. Then calibrate a head with calibration points at the program’s rate. Then use a head of Device Classes revision 0.1 that presents `org.openaquariumproject.calibration-points` with a point at the program’s rate. Then send a head with calibration points a dose at a rate below its lowest point.	The controller, or the doser, does not run the program on the first head, and tells the owner to calibrate the head at that rate. The second and third heads dose at the rate, and deliver each dose within their calibration. The last dose is refused.
Calibration due (8.4)	Controllers, dosers	8.4.2, 8.4.3	Set `recalibrateDays` to 1, calibrate the head, and let the clock run past a day. Then calibrate it again.	`calibration-due` is raised once the day has passed, doses continue, and the fault clears when the head is calibrated.
Calibration check at refills (8.5)	Controllers, dosers	8.5.2, 8.5.3	Fill a container with a measured volume, run doses of more than 100 mL in all, then record a refill, giving a volume left that differs from the estimate by 20 % of the volume used since the previous refill. Repeat with a difference of 5 % of the volume used.	The first raises `calibration-drift`, naming the head and the difference. The second raises nothing.

Test	Applies to	Requirements	Method	Pass criteria
Dose times and volumes (9.4)	Controllers, dosers	9.3, 9.4.1, 9.4.2, 9.4.3, 9.4.4	Run a program of 12 doses from 08:00 to 20:00 every 2 days, one of 3 doses at listed times on Mondays and Thursdays, and one of 24 doses from 00:00 without `end`, through a change to and from daylight saving time. Run the same programs on a head and on a controller.	Every dose falls due within 1 s of the time Section 9.4 gives, on the right days, with the right volume, within the head's calibration, and the head and the controller agree. The window without `end` lasts 23 hours on the day the clocks go forward, and 24 hours on the day they go back.
Spacing between heads (9.6)	Controllers, dosers	9.6.2, 9.6.3, 9.6.4	Put two heads in one group with a gap of 900 s, and give both a dose at 10:00. Then hold the first head from 09:55 to 10:20.	In the first case, the first head doses at 10:00 and the second no sooner than 900 s after the first dose ends. In the second, the second head doses at 10:00 and the first no sooner than 900 s after that dose ends. The doses never overlap.
Ratio dosing (9.7)	Controllers, dosers	9.7.3, 9.7.4, 9.7.6	Make a head follow another with a factor of 1.5. Change the leader's amount; cut a leader dose short by stopping it halfway; then hold the leader past a dose's maximum delay.	Each follower dose is 1.5 times the leader dose it follows, as delivered, within the heads' calibration, including after the change and the short dose. The follower's dose is missed when the leader's is.
Holds stop doses (10.2)	Controllers	7.6.1, 7.6.3, 7.6.4, 10.2	During a dose, start a mode that stops the head, then end it. Repeat with a `block` interlock and with a stop override.	The head stops within 1 s each time and starts no dose while held. After the hold, the rest of the interrupted dose is given, if within its maximum delay.
Delays and missed doses (10.3)	Controllers, dosers	10.3.1, 10.3.2, 10.3.3	With `maxDelay` 3600 s and doses 2 hours apart, hold a head for 30 minutes from a dose's due time; then for 90 minutes; then for 3 hours.	After 30 minutes, the dose is given within 1 s of the hold ending. After 90 minutes, it is missed, `dose-missed` is raised, and nothing extra is given. After 3 hours, the first dose is missed and the second is given when the hold ends, with no more than its own volume.
Catching up (10.4)	Controllers, dosers	10.4.2, 10.4.3, 10.4.4, 10.4.5	With `catchUp` and 12 doses a day, make the fifth dose missed. Separately, make the last dose of a day missed, and make the first four doses of a day missed when only two remain. Separately, make the fifth dose missed, then the sixth, which has caught up part of the fifth.	In the first, the seven remaining doses each grow by a seventh of the missed volume, and the day's total equals its amount. In the second, nothing is added to the next day. In the third, each remaining dose is at most twice its planned volume, and the rest is missed. In the fourth, no dose grows to more than twice its planned volume before catching up.
Never twice (10.5)	Controllers, dosers	7.7.3, 10.5.1, 10.5.2, 10.5.3, 10.5.4, 10.5.5, 15.2.5	Remove the controller's power halfway through a dose, then restore it. Repeat with the head's power, and with a head that runs its own program.	The interrupted dose is not given again, nor is what was left of it, and the day's record shows the volume the head delivered. Later doses are not enlarged by more than catch-up allows.
No doses without the time (10.2)	Controllers, dosers	7.5.4, 10.2	Restart the controller, or the head, with its clock not valid, across a dose's due time. Then make the clock valid, once within `maxDelay` of the due time and once after it.	No dose starts while the clock is not valid. Once it is valid, the dose is given in the first case and missed in the second.
Never a day twice (10.5)	Controllers, dosers	7.5.6, 7.5.7, 7.5.8, 7.5.9, 10.5, 14.2.4, 17.3.2	During a dosing day, after some doses are given, step the clock back by an hour. Then, after midnight, step it back into the previous dosing day. Then change the	No dose is given twice, and no dose starts until the clock again shows the time before the step. The earlier dosing day is not

PRE-DRAFT

Test	Applies to	Requirements	Method	Pass criteria
			time zone so that the local date goes back by a day. Repeat each on a head that runs its own program, and cut the head's power across midnight. Then set the clock 3 days ahead, let a dosing day run, and set the clock back to the right time; then confirm the time as a person.	planned again. The head keeps `dosedToday` across the power cut, and does not reset it twice within 23 hours, whatever its clock does. While the clock waits, the controller shows which heads wait, and until what time. After the step of 3 days, the controller asks a person to confirm the time; once the person does, no dose is given on the confirmed date, dosing resumes on the next, on the controller and on the head that runs its own program, and no dose is given twice.
Primes and calibration runs (10.6)	Controllers, dosers	8.2.4, 10.6.3, 10.6.5, 10.6.6, 14.2.6	Configure a mode with a `prime` command for a head. Ask the controller, as a person, for primes of 40 s and 30 s on a head with a `maxDailyPrimeTime` of 60 s and a `maxContinuousRun` of 600 s, and for a calibration run longer than `maxContinuousRun`. Then hold the head with an interlock, and ask for a prime and a manual dose. Read the head's description.	The configuration is rejected. The first prime runs; the second is refused, by the head or by the controller, with `command-refused`. The controller does not send the long calibration run, and tells the person the longest it can be. While the interlock holds the head, the controller sends neither. The description gives `maxDailyPrimeTime` for a head that implements `prime`.
Container forecasts (11.2)	Controllers, dosers	11.1, 11.2	Set a container's capacity, run doses and a prime, then record a refill. Set `lowDays` above the forecast.	The estimate falls by each dose's volume and by the volume primed. `low` is raised at the threshold. The refill resets the estimate.
Empty containers (11.3)	Controllers, dosers	11.3	Make a container's empty condition true, then unknown, then false. Separately, with `stopWhenEmpty`, run the estimate down to zero.	While the condition is true or unknown, and while the estimate is at zero, no dose starts, `empty` is raised, and the doses are missed. Dosing resumes when the condition is false or a refill is recorded.
Product life (11.5)	Controllers, dosers	11.5	Set `shelfLifeDays` to 1, record a refill, and let the clock run past a day. Then record another refill.	`expired` is raised after the day, doses continue, and the refill clears it.
Planning the day (12.5)	Controllers	12.5	Give a head the table of Section 12.2 and a result between two points; then a step method, a result below the target, and no new result the next day.	The table gives the interpolated amount. The step method changes the amount by `gain` times the error on the first day, and not on the second. Amounts change only when a dosing day starts.
Owner's limits (12.6)	Controllers	12.6.1, 12.6.2	Give results that ask for an amount above `max`, one below `min`, and one that asks for a change of three times `maxStep`.	The planned amounts are `max`, `min`, and the previous amount plus `maxStep`. `dose-adjust-limit` is raised for the first two.
Plausible results (12.4)	Controllers	12.4	Give a result outside the plausible range. Then a result that jumps by twice `maxJump`, followed by a second like it. Then a jump followed by a result back near the earlier ones. Then give the head `plausible` and an input of an analyzer that a testing function drives, and a result that jumps by twice the head's `maxJump`, which the testing function accepts once its retest confirms it.	The first is not used, and `implausible-result` is raised. The jump is not used until the second result confirms it, and then both are. The lone jump is never used. The result that the testing function accepted is used for the next dosing day, and the head raises no `implausible-result` for it.
Stale and missing results (12.7)	Controllers	12.7, 12.7.7	Let the latest result grow older than the input's `maxAge` before a dosing day starts, with each `fallback`. Then disconnect the	Each day is planned as its `fallback` says, and `input-unavailable` is raised. When a

PRE-DRAFT

Test	Applies to	Requirements	Method	Pass criteria
			analyzer. Then give the head, with `fallback last`, accepted results of an analyzer whose reagent's `expires` date is 29 days past, and then 31 days past.	result is accepted again, the next day is planned from it, within `maxStep` of the day before. At 29 days, the day is planned from the result; at 31 days, it is planned at the normal amount, and recorded with the basis `expired-reagent`.
Pausing on a condition (13.1)	Controllers	13.1	Pause a head while the pH is above 8.45. Raise the pH across a dose's due time, then make the pH reading unavailable across another. Give the head a follower.	Both doses are skipped, and neither is caught up later. The follower's doses are skipped with them. `input-unavailable` is raised for the pH reading.
Doses into moving water (13.2)	Configurations	13.2.2, 13.2.3	Stop the return pump with a feed mode across a dose's due time, for less than `maxDelay`.	No dose starts while the pump is stopped, or until the interlock's delay after it restarts. The held dose is then given.
Limits on the head (14.2)	Controllers, dosers	14.2.1, 14.2.2, 14.2.3, 14.2.5	Bind a program to a head without `maxDailyVolume`. Then set it, and send `dose` commands, from the controller and from a test tool, that would exceed each limit.	The controller does not run the program on the first head, and says why. The head refuses every dose that would exceed a limit, from any source, and stops a dose in progress at the daily limit.
The day's amount (14.4)	Controllers, dosers	14.4.1, 14.4.2, 14.4.3	Run a day with delays, catch-up, interruptions, a restart, and a change to the program partway through.	The program's doses on the head add up to no more than the day's planned amount, within the head's calibration.
Unexpected dosing (14.5)	Controllers	14.5.1, 14.5.2, 14.5.3	Make a simulated head report `dosing` for 10 s with no dose sent. Separately, raise its `dosedToday` by 5 mL more than the doses sent. Separately, prime the head for 10 s, and make it report `dosing` for 10 s after the prime should have ended. Separately, make the head report `controlSource local` and `dosing` for 10 s, with a `doseCompleted` event of 5 mL, then end the local control. Then, with a head that runs its own program with a normal amount of 50 mL a day, raise its `dosedToday` to 55 mL within a local date; and make it report `dosing` for 10 s while a hold without `skip` holds it.	Each time, the controller sends `stop` within 1 s of the threshold, raises `stuck-on`, and starts no dose on the head, across a restart, until a person clears it, except under local control: the dose a person ran at the head is not taken for unexpected dosing, then or after the local control ends. The prime itself is not taken for unexpected dosing.
Dosing heads only (14.6)	Controllers, configurations	14.6.1	Configure a dosing function whose head is a `switch` output.	The configuration is rejected, with the reason.
Duty limits (14.7)	Controllers	14.7.2, 14.7.3, 14.7.4	Give a head with a `maxContinuousRun` of 60 s a dose that takes 150 s at its rate. Then ask the controller for a prime of 90 s.	The dose is given in three parts of at most 60 s, with rests of at least the length of the part before, and its volume is the dose's volume. The controller does not send the prime of 90 s, and tells the person the longest it can be.
The same rules on a head (15.2)	Dosers	15.2.1, 15.2.6, 15.2.7	Run the tests for dose times, spacing, ratio dosing, delays, catching up, never twice, empty containers, and the day's amount on a doser with no controller connected.	Each passes as it does on a controller.
Holding a head's program (15.3)	Controllers, dosers	15.2.3, 15.2.5, 15.3.3, 15.3.4, 15.3.5, 17.5	Start a 10-minute feed mode that stops a head running its own program, then stop the controller's evaluations during the mode. Repeat with a feed mode that stops only the return pump, which the head requires with a `delay` of 60 s. Then, with `failSafeAction state`, hold the head with an emergency mode, stop the	The head stops within 1 s and starts no dose. It resumes its program when the hold's `revertAfter` falls due, and gives the held dose if within its maximum delay; held through the return pump, it resumes 60 s after the feed mode would have ended.

Test	Applies to	Requirements	Method	Pass criteria
			controller's evaluations, cut the head's power, and restore it. Then, with `failSafeAction continue`, hold the head with a pause condition, and stop the controller's evaluations. Then bind a head of Device Classes revision 0.1 that implements `org.openaquariumproject.set-program`, and one that implements neither it nor `set`, each with `runOn head`.	The head held by the emergency mode is sent a latched hold, and stays held, after its `failSafeTimeout` and before and after the power cut, until the controller returns and the mode ends. The head held by the pause condition stays held until a person ends the hold. The head of revision 0.1 is held with `org.openaquariumproject.set-program`; the controller runs the program of the last head itself.
Adjusted amounts expire (15.3)	Controllers, dosers	15.3.6, 15.3.7	Give a head that runs its own program a measurement-driven amount, then stop the controller for two days.	The head doses the adjusted amount on the day it was written for, and its normal amount on the days after.
Writing head programs (15.3)	Controllers	15.3.1, 15.3.2, 15.4.1	Change a head's amount and its group in the configuration. Then change the head's program on the head itself. Then remove the configuration's `posixTimeZone`. Then set the head's `runOn` to `controller`, and let it dose across local midnight on a module bus whose network time is UTC.	Within a minute, the head's `program` holds the configuration's head program, without whitespace, and no longer than 256 bytes on the module bus. Without the configuration's `posixTimeZone`, the program still carries a `posixTimeZone`, which gives the same local times as the configuration's `timeZone` for at least a year. With `runOn controller`, the head's `program` holds only `posixTimeZone`, the head follows the controller's `dose` commands, and its `dosedToday` starts again at local midnight.
Invalid configurations	Controllers, configurations	7.2, 7.2.3, 7.2.4, 7.2.6, 9.6.5, 9.7.1, 9.7.2, 10.3.6, 10.6.5, 12.2.2, 12.2.3, 12.3.3, 12.5.2	Import configurations that each break one rule for documents: a dosing function whose vessel belongs to another system; an output with no head; a dosing function with `onUnavailable`; a function that fails its schema; a spacing group with heads on the controller and heads that run their own programs; a follower whose leader follows another head; a follower with an `amount`; a `maxDelay` of more than a day; a mode with a `prime` command for a head; a normal amount outside `min` and `max`; a follower with `adjust`; an `observation` input without `maxAge`; and table points out of order.	The controller and the validator reject each one, saying why, and the running configuration is unchanged.
Control process	Controllers	7.1.3	Stop the controller's control process while its communication keeps working, while heads that it runs are idle and while one doses.	The heads stop receiving supervision messages, and apply their fail-safe behavior once their `failSafeTimeout` has passed; no dose starts meanwhile.
Saved state	Controllers	7.2.8, 7.7.1, 7.7.3	Read the saved state every second while a day's doses run, are held, and are caught up. Remove the controller's power just before a dose is sent, and again just after. Export the configuration, and import it on another controller. Then import a configuration of Automation revision 0.2 whose dosing function has the type `org.openaquariumproject.dosing`, with its saved state under `state.extensions`, and export it.	The saved state is valid against `$defs/savedState`, changes within 1 s of each change, and records each dose as `dosing` before the head reports it. After each restart, and on the other controller, the day continues from the saved state, and no dose is given twice. The configuration of revision 0.2 continues from its saved state, and is exported with the type

Test	Applies to	Requirements	Method	Pass criteria
				`dosing`, and the saved state in `state.functions`.
Visibility	Controllers	7.9	With heads that have `adjust`, a container, a spacing group, and a program that runs on the head, let a day run with a dose held by a maintenance mode and a dose missed. Then step the controller's clock back by an hour.	The controller shows each item of Section 7.9 for each head, including why the held dose is delayed, and, after the step, that each head waits, and until what time.
Head programs	Dosers	15.1.2, 15.1.4, 15.1.6	Write to a head's `program`: a valid head program; one that fails `$defs/headProgram`; one with a member the head cannot run; valid programs whose `posixTimeZone` has rules in the `M`, `J`, and plain-number forms, with transition times; and a program with only `posixTimeZone`. Run each across its transitions.	The head accepts the valid programs and rejects the others, keeping its previous program. Its doses fall due at the local times each time zone gives, within 1 s, across each transition. With the program of only `posixTimeZone`, it starts no dose of its own, follows `dose` commands, and resets `dosedToday` at the local midnight that time zone gives.
Records	Controllers	16.1.3	Let a controller that records a head's given doses as one total a day run a day of 24 doses in which two doses are missed and one is skipped, with a manual dose. Repeat with a window that crosses midnight, writing a Data Model document of revision 0.4, and then of revision 0.2.	The total has `occurrences`, `time`, `end`, and the `amount` of the given doses; the missed and skipped doses are recorded one by one, and the manual dose on its own. The day whose window crosses midnight is recorded as one total in the document of revision 0.4, and as one total for each date in the document of revision 0.2.
Imported configurations	Controllers	20.7	Import a configuration with a dosing function.	Before it runs it, the controller shows the owner the amounts, limits, and products of each head, and runs it only once the owner confirms.

20 Security Considerations

1. **Dosing can poison a system.** Changing a dosing function needs the `change-configuration` operation, and changing a head's limits `change-device-settings` [OAP-SA Table 6], both of the `full-control` role.
2. **Manual doses add chemicals.** A manual dose, a prime, or reversing a head needs `dose-now`, of the `full-control` role, not the `override` role that can start a feed mode [OAP-SA Table 6]. No automation sends a prime or a calibration run (Section 10.6.5), and a head limits them by itself, so a person with that role primes only within `maxDailyPrimeTime`.
3. **Calibration changes every dose.** Calibrating needs `calibrate`, of the `full-control` role. A controller SHOULD show how much a calibration changes the flow rate, and ask for confirmation of a change of more than 10 % (Section 8.2.5).
4. **Refills.** Recording a refill needs `record-maintenance`, of the `override` role [OAP-SA Table 6]: a wrong refill hides an empty container, but adds nothing to the water.
5. **Clearing `stuck-on` from afar.** A person who clears it remotely lets a head dose again that nobody has looked at. A controller SHOULD say so, and ask the person to confirm that someone has checked the head.
6. **Results steer doses.** Anyone who can forge a result, or enter a manual test, can change the day's amount. `min`, `max`, `maxStep`, and the plausibility checks bound what a false result can do (Section 12), and network bindings authenticate devices [OAP-SA Section 10.3]. Entering test results, or importing laboratory documents, needs `enter-results`, which only the `full-control` role permits, so that an owner grants it on purpose [OAP-SA Table 6].

7. **Imported programs.** A program from another owner can set amounts that do not suit the system it is imported into. A controller MUST show the owner the amounts, limits, and products of every imported dosing function before running it, as Automation Section 21, item 2, requires for imported automation.
8. **Radio links.** A head reached by radio stops on loss of contact, or keeps to its own program within its own limits and keeps the holds its controller sent (Section 17.1).
9. **Clocks and time zones.** Anyone who can set a device's clock or time zone could try to make a day fall due twice. A dosing day is never planned twice, a clock that steps back is not valid for dosing until it catches up, or, after a step of more than a day, until a person confirms the time (Section 7.5), and a head resets `dosedToday` at most once in 23 hours (Section 14.2.4). Confirming the time needs `change-configuration`, of the `full-control` role [OAP-SA Table 6], and dosing then resumes only from the next date.
10. **Privacy.** Dose records show what an owner keeps and how, and manual doses show when someone is at home; they SHOULD be treated as the owner's data.

Appendix A — Approval Checklist

Complete this checklist before requesting Working Group approval. Link each item to the evidence.

Item	Status	Details
Section 2 describes compliance with all five tenets.	Done	
A public review period of at least 30 days has been announced and completed.	Not started	
At least two independent implementations exist, at least one from an organization not represented among the Authors.	Not started	
The implementations have passed the test procedures in Section 19.5, including a doser that runs its own programs and a controller.	Not started	
A head program written by one implementation has run on a doser of another.	Not started	
Every normative reference is publicly available.	Done	
The checks file, schema, and examples are published, and the examples pass the validator.	Done	
Experienced aquarists have reviewed the recommended values and daily changes.	Not started	
The Open Issues appendix is empty, or each open issue is explicitly deferred.	Not started	
No unresolved Exclusion Notices for this Specification are recorded in `Notices.md`.	Done	
The Maintainer has recorded consensus to approve.	Not started	

Appendix B — Examples (Informative)

The `examples` directory contains:

- `reef-two-part.json`: a reef with a four-head doser module run by the controller: alkalinity adjusted from an analyzer by the step method, and paused while the pH is high; calcium following alkalinity in a ratio of 1 to 1; magnesium planned from the owner's tests by the table method; and trace elements every second day; with spacing, catching up, containers, an interlock on the return pump, and the saved state of a day on which a maintenance mode made two doses miss.
- `planted-fertilizers.json`: a planted freshwater tank with a three-head network doser running its own programs: macronutrients adjusted from the owner's nitrate tests, micronutrients spaced an hour from them, and liquid carbon with an empty sensor in its container.
- `doser-head-programs.json`: the device description of a two-head doser of Device Classes revision 0.5 running a two-part supplement by itself, with its head programs and its limits.
- `history.json`: a Data Model document of revision 0.2 with dose records, given, caught up, missed, and manual, a day's total, an adjustment, a calibration, and a refill.
- `minimal.json`: one head, one dose a day.

Appendix C — Commercial Features (Informative)

How the dosing features of commercial controllers correspond to this Specification.

Table 23 — Commercial features

Feature	In this Specification
Calibration	Section 8; at several speeds, Section 8.3.
Daily total split across a time window, by weekday or every N days	Sections 9.3 and 9.4.
Spacing doses between heads	Section 9.6.
Container tracking, low-level alerts, a days-left forecast, and stopping when empty	Section 11.
Catching up missed doses	Section 10.4.
Dosing adjusted from measurements, interpolated, with a fallback	Section 12: the table method, and `fallback`.
Fixed-ratio multi-part dosing	Section 9.7.
Stopping doses on a condition, such as high pH	Section 13.1.
Duty-cycle protection for the pump	Section 14.7.
Water changes using dosing pumps	Water Level and Water Change Section 9.12.

Appendix D — Open Issues (Informative)

Issues 1 to 24 were open in revision 0.1. The Project Lead decided issues 22, 25, and 26 on 15 October 2026, and issues 1 to 7 and 24 on 16 October 2026. Issues 27 and 28 arose in revision 0.2, and the Project Lead decided them on 16 October 2026, accepting the recommendation in each. On 17 October 2026 the Project Lead froze the scope of revision 1.0, and decided issue 12; revision 0.3 applies the decision, applies issue 27, and resolves issues 8, 13, and 14, which other specifications settled. Issue 29 records the choices that revision 0.3 made where those decisions left them open, which the Project Lead accepted on 18 October 2026. Issues 16 to 21 and 23 are deferred beyond revision 1.0, and are tracked outside this Specification. Issue 15 stays open until the bench tests and owners' experience confirm its numbers. Each issue says how a revision resolves it, or why it stays open. Issue numbers do not change, because other specifications refer to them.

Questions for the Project Lead

1. **Default limits for measurement-driven dosing.** *Decided by the Project Lead on 16 October 2026; resolved in revision 0.2: no silent defaults.* `min`, `max`, and `maxStep` have no defaults, and a controller must not run measurement-driven dosing until the owner has set them (Section 12.2). The alternatives are defaults as a share of the normal amount, as commercial controllers bound their adjustments, at 20 % or 35 % to 40 % of the normal dose, with a `maxStep` of 10 %, or defaults in terms of the water, computed from the product's effect, such as a `maxStep` that changes alkalinity by no more than 0.25 dKH a day. Defaults are convenient, but a share of the normal amount means very different things for a nano reef and a large one, and effects are only as good as the maker's figures. The Project Lead decided on no silent defaults: a controller proposes values from the product's effect and Table 13, and the owner confirms them (Sections 12.2.1 and 12.8.2).
2. **A shared registry of dosing products.** *Decided by the Project Lead on 16 October 2026: deferred.* Products are named by free text and a GTIN, and their effects are given by each owner (Section 12.8), as Data Model products are [OAP-DM Section 8.6]. A shared registry, with an identifier, the maker, the parameters each product changes, its effect, its ratio to its other parts, and its shelf life once opened, would let controllers propose gains, limits, ratios, and spacing, and make histories comparable between owners. Against it: makers change formulations, an effect is the maker's claim, someone must maintain the registry and check its sources, and listing products may look like endorsing them. The Project Lead decided to keep the product format of this revision, `product` and `effect`, and to decide on a registry once the Data Model's GTINs have been used for a while; a registry could then be a separate data set, open to contributions, with a source for every figure.

3. **Fallback when results stop.** *Decided by the Project Lead on 16 October 2026; resolved in revision 0.2: `normal` stays the default.* The default `fallback` is `normal`, as Fail-Safe Behavior Section 10.3.2 suggests (Section 12.7). After months of measurement-driven dosing, the normal amount may be far from what the system uses, and `last` may be closer; but `last` keeps dosing an amount chosen from results that are no longer checked. Commercial controllers fall back to the normal dose. The Project Lead decided to keep `normal`, which no unchecked result has moved, and to keep the normal amount close to what the system uses instead: a controller proposes a new normal amount when the amounts planned from results have stayed far from it (Section 12.7.6).
4. **Catching up by default.** *Decided by the Project Lead on 16 October 2026; resolved in revision 0.2: off by default, proposed by product.* `catchUp` defaults to `false` (Section 10.4). Catching up keeps the daily amount, which matters for supplements the system uses steadily, and never doubles a dose; skipping is simpler, and errs towards a smaller dose. One maker's supplement manuals say to make up missed calcium and alkalinity doses, but never carbon or nitrate and phosphate doses, so a default by kind of product, from a product registry (issue 2), may suit better than one default. The Project Lead decided to keep it off by default, which errs towards a smaller dose, and to have a controller propose it for alkalinity, calcium, and magnesium supplements, and not for other products (Section 10.4.8).
5. **Dosing heads only.** *Decided by the Project Lead on 16 October 2026; resolved in revision 0.2: kept.* Only `doser` outputs can be heads (Section 14.6), because only they enforce their own limits. Owners who dose with a pump on a switched outlet must add a module that presents it as a doser. The Project Lead decided to keep the restriction, because a controller's limits alone do not stop a pump whose switch fails closed (Section 14.6.2), and such a pump can be presented as a doser by a module that enforces its limits (Section 14.6.3).
6. **A follower that cannot dose.** *Decided by the Project Lead on 16 October 2026; resolved in revision 0.2: kept.* When a follower cannot dose, its leader continues (Section 9.7.8), so the parts of a multi-part supplement go out of balance until the owner acts. The alternative is to stop the leader too, which keeps the balance but stops a supplement the system may need. Which is safer depends on the product. The Project Lead decided that the leader continues, as Section 9.7.8 says, so that a supplement the system needs keeps coming, while the follower's alarms, such as `empty` and `dose-missed`, tell the owner to restore the balance.
7. **Maximum delay.** *Decided by the Project Lead on 16 October 2026; resolved in revision 0.2: kept.* `maxDelay` defaults to 1 hour. Owners with one dose a day may want most of a day, as one maker allows missed doses to be given until midnight. The Project Lead decided to keep the default of 1 hour, and an owner who wants longer can set `maxDelay` up to 86 400 s, which Table 7 allows.

For other specifications

8. **Automation.** *Resolved in revision 0.3, with Automation revision 0.3.* The Automation specification carried dosing functions under a provisional extension type, with their saved state under `state.extensions["org.openaquariumproject.dosing"]`. Automation revision 0.3 lists `dosing` among its function types, and keeps each hosted function's saved state as `state.functions["<function id>"]` (its Section 11.10 and Table 14); says that a dosing function sends `dose` to `doser` outputs, which have no running state, only while no higher layer holds them (its Section 8.3.1); forbids `prime`, `calibrate`, and `completeCalibration` as mode commands, because only a person starts them (its Section 14.5.2); and points its `doser` preset to the new type. Sections 7.2 and 7.7 now use the type and the saved state of that revision, and a controller still reads the names of revisions 0.1 and 0.2 (Section 7.2.8).
9. **Device Classes.** *Resolved in revision 0.2.* Device Classes revision 0.2 added the `calibration-points` feature and `calibrate`'s `rate`; `set` and `enabled` for heads with `local-program`; `maxDoseVolume`, `maxDailyVolume`, and `dosedToday` REQUIRED for every head; and `program` and `containerRemaining` as this Specification asked (its Section 9.7 and Appendix D, issues 1 and 18). Sections 8.3 and 15.3 now use them, and keep `org.openaquariumproject.calibration-points`, `org.openaquariumproject.set-program`, and `org.openaquariumproject.program-enabled` only for heads of revision 0.1 [OAP-DC Table 5].
10. **Data Model.** *Resolved in revision 0.2.* Data Model revision 0.2 added the event types `dose-adjusted` and `refill`, totals of a day's doses with `occurrences`, Calibration Records with the model `flow-rate`, and product GTINs (its Sections 8.6 and 8.12, and Appendix C, issue 19). Section 16 now uses them, and keeps

the provisional event types of revision 0.1 only for reading older documents (Section 16.3.3). A registry of products is issue 2, and totals over a dosing day are issue 27.

11. **Alarms and Notifications.** *Resolved in revision 0.1.* `conditions.json` lists `dose-missed`, `dose-adjust-limit`, and `calibration-drift` (Table 5), as Alarms and Notifications Appendix D, issue 14, asked. Dosing raises Device Classes codes, such as `stuck-on` and `implausible-result`, with the function as their subject (Section 7.8).
12. **Security and Access.** *Resolved in revision 0.3, as the Project Lead decided on 17 October 2026, with Security and Access revision 0.3.* Recording a refill used `run-action`, provisionally, and entering a manual test result, which can drive dosing (Section 12.3), had no operation of its own. Security and Access revision 0.3 adds `record-maintenance`, which the `override` role permits, for records that change no output, such as refills, and `enter-results`, which only the `full-control` role permits, so that an owner grants it on purpose, for entering test results and importing laboratory documents (its Table 6 and Appendix D, issue 30). Section 11.4.1 and Section 20, items 4 and 6, now use them.
13. **Water Testing.** *Resolved in revision 0.3, with Water Testing revision 0.3.* The Water Testing specification screens analyzer results by the rule of Section 12.4, retests them, and corrects them, before any input reads them, and records accepted results as observations (its Sections 10, 11, 13, and 15). It proposed that this Specification take a result's status from it, rather than judge it again (its Appendix D, issue 12). Section 12.4.5 now does: a head whose input reads results that a testing function screens uses the results that the testing function accepted, and does not judge them again by its own `plausible`, so that a result that a retest has confirmed is not held back again until the next test. Section 12.7.7 also plans no day from a result of a test that used a reagent more than 30 days past its expiry, as the Water Testing specification requires (its Section 9.4.3).
14. **Module Bus.** *Resolved in revision 0.3, with Module Bus revision 0.4.* A head program must fit in a 256-byte register (Section 15.4), and the module bus carries no time zone. Module Bus revision 0.4 says that a module takes the local time zone from the `posixTimeZone` that its controller writes in the program it runs, and otherwise uses its own setting (its Section 9.4.9). A head that its controller ran was given an empty `program`, and so no time zone, so it never learned when local midnight was, never reset `dosedToday`, and refused doses once `dosedToday` reached `maxDailyVolume`. From revision 0.3, a controller writes such a head a program whose only member is `posixTimeZone` (Sections 15.1.1 and 15.3.2), and tells the owner to set the time zone of a head on the module bus without the `local-program` feature, which only its own setting gives (Section 14.2.7).

Details of this Specification

15. **Numbers.** *Open until the bench tests of the breadboard prototype, and owners' experience, confirm them.* The calibration volume and drift, the smallest dose, the gap, the maximum delay, the catch-up limit, `lowDays`, the unexpected-dosing thresholds, the head-limit margins, the duty rests, and the daily changes of Table 13 need confirming on the breadboard and with owners. So do the numbers that revision 0.2 adds: the 60 s step back of the clock after which dosing waits (Section 7.5.7), and the 23 hours between resets of `dosedToday` (Section 14.2.4). So do the 20 % and 7 dosing days after which a controller proposes a new normal amount (Section 12.7.6). The Project Lead decided on 16 October 2026 to keep the numbers that revision 0.2 adds, provisional, with the others. Revision 0.3 adds the 24 hours of a step back of the clock after which a person confirms the time (Section 7.5.9), and uses the 30 days after a reagent's expiry, from which its results adjust no dosing, of the Water Testing specification (Section 12.7.7) [OAP-WT Section 9.4.3].
16. **Calibration by running time.** *Decided by the Project Lead on 15 October 2026, with Device Classes Appendix D, issue 28; resolved by Data Model revision 0.3.* Tubing wears with running time rather than days, and one maker advises calibrating every 100 hours of running. Data Model revision 0.3 lets a Task fall due after a number of hours of running, its `everyHours`, counted from the `operatingHours` of the outputs that drive its equipment, which a controller can use to remind the owner to calibrate a head (Section 8.4.1).
17. **Certification.** *Decided by the Project Lead on 16 October 2026; resolved: yes.* Should passing the test procedures of Section 19.5 be required for a future compatibility mark? The project's governance says that a compatibility program, if created, will require conformance testing (its Section 5), and the tests of Table

22 are those of this Specification, as Fail-Safe Behavior Appendix C, issue 7, says of its own. Which products may carry a mark, and how their results are checked, belong to the program, not to this Specification.

Decided by the Project Lead

25. **Heads that run their own program, after power-up.** *Resolved in revision 0.2.* Table 19 gives `fail-safe` after power-up for heads that run their own program, as Fail-Safe Behavior Table 3 does for `doser`: with `continue`, a head runs its program again after power-up, unless a hold that survived the power cut stops it. A head keeps its hold, `enabled`, across a power cut (Section 15.2.5), and a dose that a power cut interrupts never resumes (Section 10.5), as Device Classes Appendix D, issue 35, decided. With `continue`, a head keeps the holds its controller sent when it loses contact, so a hold lasts until its `revertAfter` falls due, or until a person ends it (Sections 15.2.3 and 17.5).
26. **Primes and calibration runs.** *Resolved in revision 0.2.* Controllers send `prime` and `calibrate` only when a person asks, and no mode sends them (Section 10.6.5). A head refuses a prime or a calibration run longer than its `maxContinuousRun`, and a prime past its `maxDailyPrimeTime` (Section 10.6.6), as Device Classes Appendix D, issue 37, decided.

The following issues arose in revision 0.2:

27. **Data Model: totals over a dosing day.** *Decided by the Project Lead on 16 October 2026; resolved in revision 0.3, with Data Model revision 0.3.* Data Model revision 0.2 totals the occurrences of an event within one local day (its Section 8.6.2), but a dosing day whose window crosses midnight spans two local dates (Section 7.5.1). Data Model revision 0.3 lets a total cover a day as the specification of its equipment defines it, such as a dosing day, or a testing day of the Water Testing specification. Section 16.1.3 now records a dosing day as one total, whatever the dates it spans, in documents of Data Model revision 0.3 or later, and as one total for each date in documents of revision 0.2.
28. **The leak lockout and dosing.** *Decided by the Project Lead on 16 October 2026; resolved in revision 0.2: kept (Water Level and Water Change Appendix D, issue 25).* A leak stops a head only through an emergency mode or an interlock that the owner configures (Section 13.2.4). The leak lockout stays limited to the equipment that moves water.

The following issue records the choices of revision 0.3:

29. **Choices of revision 0.3.** *Resolved in revision 0.3: the Project Lead accepted each choice on 18 October 2026.* Revision 0.3 made these choices where the decisions it applies, and the findings it resolves, left them open:
    - after its clock steps back by more than 24 hours, a controller asks a person to confirm the time, which needs `change-configuration`, and then plans no dosing day before the date after the confirmed date, so that a day that may have been dosed under the wrong time is skipped rather than dosed twice; a head that runs its own program takes a program written to it after such a step as the same confirmation (Sections 7.5.9 and 20);
    - a head program whose only members are `posixTimeZone` and `extensions` makes a head follow `dose` commands while it knows local midnight, so that a controller gives the time zone to every head with the `local-program` feature that it runs (Sections 15.1.1 and 15.3.2);
    - a day planned at the normal amount because its latest result came from a reagent long past its expiry is recorded with the basis `expired-reagent` (Sections 12.7.7 and 16.2.1);
    - the Water Testing specification is a normative reference, because Sections 12.4.5 and 12.7.7 rely on it (Section 4.1);
    - an `observation` input reads screened results when a testing function of its system checks its parameter (Section 12.4.5);
    - a head that runs its own program doses unexpectedly when its `dosedToday` rises by more than the planned amounts of the dosing days that overlap the date, the manual doses, and the doses under local control, plus 1 mL and 5 % of that total (Section 14.5.3); and
    - a head of Device Classes revision 0.4 or earlier without `maxContinuousRun` is limited only by its other limits, with no rule of this Specification's own (Section 14.7.1).