



OPEN AQUARIUM PROJECT SPECIFICATION

Water Testing

PRE-DRAFT

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

Pre-Draft. This is a proposed draft. The Working Group has not approved it, and it will change. Comment by opening an issue with the `water-testing` 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, starting points 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, starting points file, schema, and examples under the Apache License 2.0. Every requirement it makes has a published test procedure that names it (Section 19.5). The schedule format, the rules for judging a result, and the arithmetic of corrections are published in full, with a validator that includes a reference implementation of each, so that owners can see why a result was used or not, and makers can check that their products agree.

2.2 Interoperability

One schedule format runs the same way on a controller and on an analyzer that runs its own schedule (Section 14), so an analyzer from any maker works with any controller. Testing is a function type of the Automation specification, so it moves between controllers with the rest of an owner’s configuration. It uses the Device Classes `analyzer` class, Automation inputs, conditions, and modes, Data Model parameters, units, observations, samples, events, and calibration records, Fail-Safe Behavior terms, and Alarms and Notifications condition codes, and adds no parallel vocabulary. Results from analyzers, manual tests, and laboratories are recorded in one form, so that dosing, alarms, and charts use them alike (Sections 12 and 15).

2.3 Local Control

Schedules, screening, retests, and corrections all run on the controller or on the analyzer, with no network service, account, or vendor. Today, an analyzer’s results reach another maker’s equipment mostly through the makers’ cloud services, or through an analog output that imitates a pH probe; this Specification needs neither. Owners keep every result, test, comparison, and correction in their own history, and can export it.

2.4 Safety

This Specification exists so that a wrong result does no harm. Results steer dosing, and an analyzer with an air bubble in its sample, an empty reagent, or a worn pump can report a value that looks real. Every result is screened before anything uses it, and a result that looks wrong is tested again; only results that pass reach dosing, alarms, or the owner's history as observations (Sections 10 and 13). A result whose chemical basis was only assumed, or whose sample time is later than the controller's clock, is used only once a person has confirmed or corrected it (Section 12.7). An analyzer whose reagent is known to be empty does not test, and its results are not believed (Section 9.3). A controller warns its owner about an analyzer whose empty reagents or full waste nobody can detect, because then neither is known (Sections 9.3 and 9.5). An analyzer that keeps failing stops, and results that stop arriving raise a fault, so dosing falls back, by default to its normal amounts (Sections 10.5 and 12.6). Results of a reagent more than 30 days past its expiry no longer adjust dosing, while alarms still see them (Section 9.4). Plausibility limits and correction limits start from published values for some systems, but a controller applies none until its owner confirms it (Sections 10.2 and 11.6). Corrections need a person, are applied exactly once, and raise a fault when they are larger than the owner allows; without the owner's limit, a person confirms each correction of more than 10 % (Section 11). Water that has met any liquid the analyzer adds, its cleaning solutions included, goes to waste, not back into the system, unless its maker shows that returning it is harmless (Section 9.5).

2.5 Sustainability

Reagent forecasts let owners order consumables before they run out, and stopping a failing analyzer saves reagent and waste room for when it is fixed (Sections 9.2 and 10.5). Comparisons with laboratory results let owners keep using an analyzer whose results have drifted, with a recorded correction, rather than replace it (Section 11). Because schedules are portable and results use the Data Model, owners keep their analyzers and their history 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	A held analyzer that keeps going when it loses contact stays held until its hold's <code>revertAfter</code> falls due, and a controller tells a held analyzer by its <code>enabled</code> , and by its <code>state</code> only for an analyzer of Device Classes revision 0.2 or earlier; the analyzer members of Device Classes revisions 0.2 and 0.3: <code>raw</code> , <code>testEnded</code> , <code>methods</code> , <code>reagents</code> , <code>tests</code> and <code>parameters</code> , <code>wasteTests</code> , tests of a standard, and the <code>unit</code> of a correction's offset; history recorded with the Data Model's own objects: the event type <code>analyzer-test</code> , an observation's <code>correction</code> and <code>reference</code> , and Calibration Records for comparisons and corrections; results whose basis was only assumed, and results whose sample time is later than the controller's clock, used only once a person has confirmed or corrected them; every liquid the analyzer adds, cleaning solutions included, kept out of the system; a warning about analyzers whose empty reagents or full waste nobody can detect; results taken only once while an analyzer's clock is not valid; skipped tests counted as failures unless a hold caused them; at least three comparisons for a proposed correction, and a proposal that stops counting once a person acts on it; an analyzer program that is not loaded raises <code>program-not-loaded</code> and holds the analyzer; the time zone always written to analyzer programs; stale results and inputs' maximum ages reconciled; <code>test-due</code> raised and cleared by accepted results; the section on sampling made normative; a test procedure, mapped to its requirements, for every requirement; references cited by tag; one sentence per line; the decided issues resolved, and every other issue given a status.	Trent Scholl
0.3	2026-10-18	The Project Lead's decisions of 17 October 2026 applied: the function type <code>water-testing</code> , with its saved state as <code>state.functions["<function id>"]</code> , in place of the provisional names, which controllers still read; a published table of starting points for plausibility limits and correction limits of reef systems, in <code>starting-points.json</code> , applied only on the owner's confirmation, with <code>maxJump</code> proposed from the test interval; without the owner's <code>maxCorrection</code> , a person confirms each correction of more than 10 %; from 30 days after a reagent expires, its results no longer adjust dosing; a suspension ends when a person clears it, or a test on demand gives an accepted result,	Trent Scholl

Revision	Date	Description	Authors
		or, when only skipped tests caused it, a replaced reagent or an emptied waste container is recorded; the analyzer's own calibration recorded as a Data Model Calibration Record without a result; replaced reagents and emptied waste recorded with <code>record-maintenance</code>, and manual and laboratory results entered with <code>enter-results</code>; results that a module kept while its controller was away read from it; the schema's <code>\$id</code> https://openaquariumproject.org/schemas/water-testing/0.3; returning tested water, correction by factor, corrections only by a person, warnings about consumables that nothing tracks, and holding an analyzer whose program does not load, no longer provisional. Also: the saved state's failures in a row kept for each parameter, as Section 10.5 counts them; a test window without <code>end</code> ends at the same local time on the next date, as a dosing window does; <code>program-not-loaded</code> defined with no time in it, and the analyzer held once it has lasted its delay; the time zone of an analyzer program derived as Dosing derives it; results that prove later than the clock once it is valid; a MUST behind every REQUIRED check; "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 20, 22, 23, and 25 to 27 deferred beyond revision 1.0.	

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 Schedules Run	11
7.2	The Testing Function	11
7.3	Analyzers	12
7.4	Parameters	13
7.5	Units and Times	13
7.6	Holds	14
7.7	Saved State	14
7.8	Faults and Alarms	15
7.9	Visibility	16
8	Test Schedules	16
8.1	Overview	16
8.2	Schedule Entries	16
8.3	Testing Days	17
8.4	Test Times	17
8.5	Starting a Test	17
8.6	Delays and Skipped Tests	18
8.7	Tests and Doses	18
8.8	Tests on Demand and Tests of a Standard	18
8.9	Safety Checks	18
9	Consumables	19
9.1	Reagents	19
9.2	Forecasts and Warnings	19
9.3	Empty Reagents	19
9.4	Reagent Life	20
9.5	Waste	20
9.6	Replacing Reagents and Emptying Waste	21
9.7	Safety Checks	21
10	Plausibility	22
10.1	Overview	22
10.2	Plausibility Limits	22
10.3	Screening a Result	22
10.4	Retests	23
10.5	Repeated Failures	23
10.6	Results Judged Once	24
10.7	Safety Checks	24

11	Corrections	24
11.1	Overview	24
11.2	Reference Results	25
11.3	Comparisons	25
11.4	Proposing a Correction	25
11.5	Applying a Correction	26
11.6	Limits on Corrections	26
11.7	When to Compare	27
11.8	The Analyzer's Own Calibration	27
11.9	Safety Checks	27
12	Results	27
12.1	Sources	27
12.2	Analyzer Results	28
12.3	Manual Results	28
12.4	Laboratory Results	28
12.5	The Latest Result	29
12.6	Stale and Missing Results	29
12.7	Assumed Bases and Times Later Than the Clock	29
12.8	Safety Checks	30
13	Use by Other Functions	30
13.1	Inputs	30
13.2	Dosing	31
13.3	Alarms	31
13.4	Other Functions	31
13.5	Safety Checks	31
14	Analyzers That Run Their Own Schedules	31
14.1	Analyzer Programs	31
14.2	Running an Analyzer Program	32
14.3	Under a Controller	32
14.4	Size	33
14.5	Safety Checks	33
15	History	33
15.1	Results	33
15.2	Tests	34
15.3	Comparisons and Corrections	34
15.4	Reagents and Waste	34
15.5	Records of Revision 0.1	35
16	Fail-Safe Behavior	35
16.1	Outputs	35
16.2	Lost Inputs	35
16.3	Restarts and the Clock	35
16.4	A Failed Controller	35
17	Sampling and Placement	36
18	Recommended Values	36
19	Conformance	38
19.1	Controllers	38
19.2	Analyzers	38
19.3	Configuration Documents	38
19.4	Schema and Tools	38
19.5	Test Procedures	39
20	Security Considerations	45
Appendix A — Approval Checklist		46
Appendix B — Examples (Informative)		46
Appendix C — Commercial Features (Informative)		47
Appendix D — Open Issues (Informative)		47

3 Scope

This Specification defines:

- test schedules for analyzers, run by the analyzer itself or by a controller;
- how tests are started, delayed, skipped, and held;
- the tracking of reagents and waste, with forecasts, expiry, and warnings;
- plausibility checks on every result, and retests when a result looks wrong;
- comparisons of an analyzer's results with reference results, such as laboratory results, and the corrections made from them;
- how results from analyzers, manual tests, and laboratories are taken, checked, and recorded, and what happens when they are stale or missing;
- how results reach dosing, alarms, and other functions;
- records of results, tests, comparisons, corrections, and the replacement of reagents in the Data Model; and
- a test procedure for every requirement.

This Specification does not define:

- how a controller evaluates a configuration, or the elements it hosts testing in, which the Automation specification defines;
- the `analyzer` class, which the Device Classes specification defines [OAP-DC Section 10.3];
- the chemistry inside an analyzer, or how it detects an end point;
- how dosing uses results, which the Dosing specification defines;
- how alarms are delivered, which the Alarms and Notifications specification defines;
- continuous sensors, such as pH, temperature, and conductivity probes, whose readings the Fail-Safe Behavior and `Probe Interface` specifications cover;
- the values a parameter should have; or
- how laboratories analyze samples, or the format they deliver results in, beyond the Data Model.

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](https://open-aquarium.org/specs/data-model/spec.md).
- **[OAP-DC]** Open Aquarium Project, "Open Aquarium Device Classes", Revision 0.5, [specs/device-classes/spec.md](https://open-aquarium.org/specs/device-classes/spec.md).
- **[OAP-FS]** Open Aquarium Project, "Open Aquarium Fail-Safe Behavior", Revision 0.4, [specs/fail-safe-behavior/spec.md](https://open-aquarium.org/specs/fail-safe-behavior/spec.md).
- **[OAP-MB]** Open Aquarium Project, "Open Aquarium Module Bus", Revision 0.4, [specs/module-bus/spec.md](https://open-aquarium.org/specs/module-bus/spec.md).
- **[OAP-AU]** Open Aquarium Project, "Open Aquarium Automation", Revision 0.3, [specs/automation/spec.md](https://open-aquarium.org/specs/automation/spec.md).
- **[OAP-AN]** Open Aquarium Project, "Open Aquarium Alarms and Notifications", Revision 0.3, [specs/alarms-and-notifications/spec.md](https://open-aquarium.org/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](https://open-aquarium.org/specs/security-and-access/spec.md).
- **[OAP-DO]** Open Aquarium Project, "Open Aquarium Dosing", Revision 0.3, [specs/dosing/spec.md](https://open-aquarium.org/specs/dosing/spec.md).

4.2 Informative References

- **[VIM]** Joint Committee for Guides in Metrology, “International vocabulary of metrology — Basic and general concepts and associated terms (VIM)”, JCGM 200:2012, https://www.bipm.org/documents/20126/2071204/JCGM_200_2012.pdf.
- **[OAP-DP]** Open Aquarium Project, “Open Aquarium Device Protocol”, Revision 0.3, [specs/device-protocol/spec.md](#).
- **[OAP-LP]** Open Aquarium Project, “Open Aquarium Lighting Programs”, Revision 0.3, [specs/lighting-programs/spec.md](#). Its Table 5 defines `program-not-loaded`, which its Section 16.2 raises for lights.

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, and controller, in the Fail-Safe Behavior specification [OAP-FS], such as fail-safe behavior and unavailable input, in the Automation specification [OAP-AU], such as output, input, condition, function, mode, resolved state, and saved state, in the Alarms and Notifications specification [OAP-AN], such as condition code and subject, and in the Dosing specification [OAP-DO], such as head and dosing day, have the same meaning here.

Analyzer: An endpoint of the `anaLyzer` class [OAP-DC Section 10.3], and the output that a configuration binds to it.

Testing function: An Automation function that runs the schedules of one or more analyzers, and checks the results of one system (Section 7.2).

Test: One run of an analyzer that measures one or more parameters in one sample of water. A **scheduled test** falls due by a schedule; a **retest** is asked for because of an earlier result (Section 10.4); a **test on demand** is started by a person (Section 8.8).

Standard: A solution of known value that an analyzer tests in place of the system’s water, to check itself: the `test` command with `standard true` [OAP-DC Section 10.3].

Schedule: What an analyzer does and when: a list of schedule entries (Section 8.2).

Testing day: A local date on which a schedule entry falls due (Section 8.3).

Due: A test is due from the moment its schedule says it should start.

Delayed: A test that is due but has not started (Section 8.6).

Skipped: A scheduled test that was never started (Section 8.6).

Held: An analyzer is held while no test may start on it, because of a hold of Section 7.6.

Result: A measured value of one parameter in one sample: an analyzer’s result, a manual result, or a laboratory result.

Manual result: A result a person enters, from a test kit, a photometer, or another instrument (Section 12.3).

Laboratory result: A result reported by a laboratory for a sample sent to it (Section 12.4).

Raw result: An analyzer’s result before any correction.

Corrected result: A raw result after its correction (Section 11.5).

Source: Where a result comes from: an analyzer, a manual test, or a laboratory.

Accepted result: A result that has passed screening, and may be used (Section 10.3).

Suspect result: A result held back until a later result of its source agrees with it or replaces it (Section 10.3).

Rejected result: A result that is never used (Section 10.3).

Unconfirmed result: A result whose chemical basis was only assumed, which is not used until a person confirms its basis (Section 12.7).

Fresh start: A moment after which an analyzer's first result is held back until a second agrees with it: the analyzer restarted, a reagent was replaced, the analyzer calibrated itself, or a suspension of its testing was cleared (Section 10.3).

Failure: A scheduled test that, with its retests, gave no accepted result of a parameter (Section 10.5).

Reagent: Any liquid an analyzer adds to its sample or to its own plumbing: a liquid it measures with, a standard or reference fluid it calibrates or checks itself with, or a solution it cleans with.

Waste: The liquid an analyzer discards after a test, and the container that collects it.

Reference result: A result that a person trusts more than an analyzer's, used to check it (Section 11.2).

Comparison: A reference result paired with an analyzer's raw result of the same water (Section 11.3). In the terms of the International vocabulary of metrology [VIM 2.39], a comparison is the first step of a calibration.

Correction: A factor and an offset that turn an analyzer's raw results into corrected results [VIM 2.53] (Section 11).

5.3 Notation

JSON member names, function types, condition codes, and the names of attributes, commands, events, and faults appear in `monospace`. Durations are in seconds, except members whose names end in `Days`. Units are Data Model unit codes [OAP-DM]. References to other Open Aquarium specifications name them, as in "Dosing Section 12.4", or use their tags from Section 4, as in "[OAP-DO Section 12.4]". A section number with a third part refers to a numbered item of that section: "Section 10.3.4" means item 4 of Section 10.3. Numbers marked provisional are to be confirmed with owners' experience and with analyzers on the bench (Appendix D).

6 Introduction

6.1 Purpose

Automatic analyzers draw a small sample of a system's water every few hours, add reagents, and measure alkalinity, calcium, magnesium, nitrate, phosphate, or other parameters. Their results drive dosing, and so a wrong result can change what goes into a system. Every analyzer can report a wrong result: an air bubble in the sample, a reagent that has run out or aged, a worn pump, or a dirty cell. One maker says that its analyzer reads high when its reagent runs out.

Of four widely used commercial controller ecosystems, three schedule analyzers and track their reagents and waste, two retest results that look wrong, and two let owners correct an analyzer to match a laboratory. Each defines these in its own way, and results mostly reach other makers' equipment through cloud services. This Specification defines them once, for every controller and analyzer, and adds what an owner needs to trust a result: screening, retests, recorded corrections, and a fault when results stop. It also treats manual and laboratory results, which most owners rely on, in the same way as an analyzer's.

6.2 Design Principles

1. **Screen before use.** Every result is judged before anything uses it; only accepted results reach dosing, alarms, and the owner's observations (Sections 10 and 13).
2. **Retest rather than guess.** A result that looks wrong is tested again, and two results that agree are believed (Section 10.4).
3. **Judge once.** A result is judged by one set of limits, with the same rule that Dosing uses (Section 10.6).
4. **People correct, once.** A correction needs a person, is applied exactly once, and is recorded with what it was based on (Section 11).
5. **No reagent, no result.** A test that cannot have worked gives no result that is used (Section 9.3).

- 6. **Silence is a fault.** Results that stop arriving raise a fault, and the functions that use them fall back (Section 12.6).
- 7. **The same schedule everywhere.** A schedule means the same on a controller and on an analyzer (Section 14).

6.3 Overview

Figure 1 shows how a result of an analyzer becomes something other functions use.

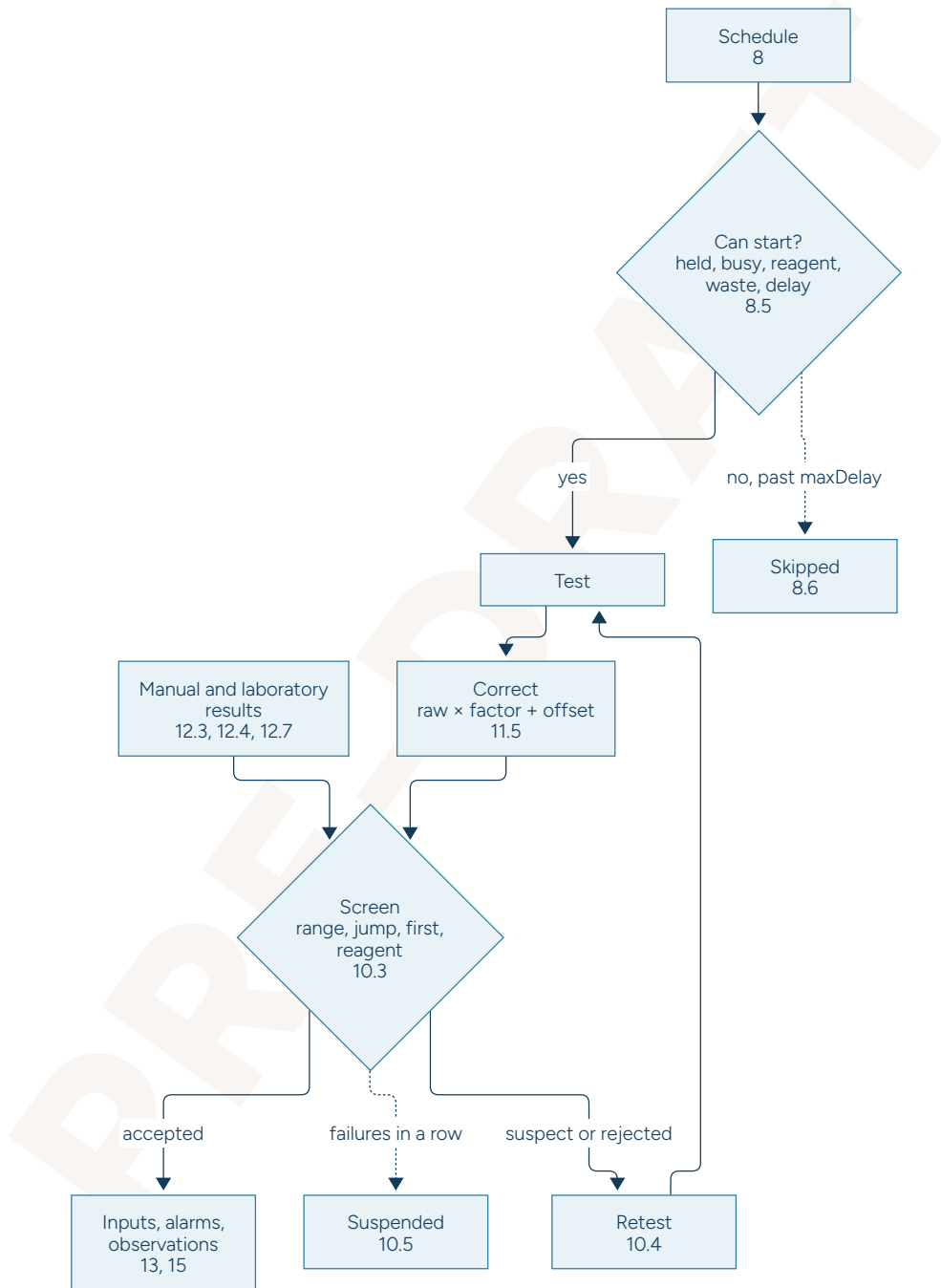


Figure 1 — From a test to a result that other functions use

Table 1 — Parts of this Specification

Part	What it does	Section
Test schedules	Say when each analyzer tests which parameters, cleans, or calibrates itself, and when a test is delayed or skipped.	8

Part	What it does	Section
Consumables	Track reagents and waste, forecast when they run out or fill, and stop tests that cannot work.	9
Plausibility	Judge every result, hold back or reject results that look wrong, retest, and stop an analyzer that keeps failing.	10
Corrections	Compare an analyzer with reference results, and correct its results, with a person's confirmation.	11
Results	Take results from analyzers, people, and laboratories, and raise a fault when they stop.	12
Use by other functions	Give inputs, dosing, and alarms only accepted results.	13
Analyzer programs	Let an analyzer run its own schedule, under a controller or without one.	14
History	Record results, tests, comparisons, corrections, and consumables in the Data Model.	15

A controller runs testing as a function type that the Automation specification hosts [OAP-AU Section 11.10]. Modes, interlocks, and overrides act on analyzers as on any other output.

6.4 Example

A reef's analyzer tests alkalinity every 4 hours, from the sump. At 08:00 it draws an air bubble with its sample, and reports 6.6 dKH, where its 04:00 result was 8.2 dKH.

1. The result is within the plausible range of 5 to 14 dKH, but differs from the last accepted result by 1.6 dKH, more than the `maxJump` of 0.5 dKH. It is held back as suspect (Section 10.3). Nothing that uses alkalinity sees it.
2. 300 s after the result, the testing function asks for a retest (Section 10.4). The retest reports 8.1 dKH, within `maxJump` of 8.2 dKH: it is accepted, and the 6.6 dKH result is rejected and recorded.
3. Had the retest also read about 6.6 dKH, the two results would have agreed, and both would have been accepted: the alkalinity really fell, and the owner's range alarms would have said so (Section 13.3).
4. Had the alkalinity reagent run out instead, no test would have started, or its result would have been rejected (Section 9.3). After three scheduled tests without an accepted result, the testing function would have stopped the analyzer and raised `testing-suspended` (Section 10.5). Only skipped tests would have caused it, so recording the new reagent would have ended it. Nine hours after the last accepted result, it would have raised `results-stale`, and the alkalinity head, whose input had become unavailable, would have planned its next day at its normal amount [OAP-DO Section 12.7].

The analyzer's calcium results that morning are corrected by a factor of 1.043, from three laboratory results that the owner confirmed (Section 11).

7 Common Requirements

7.1 Where Schedules Run

1. A schedule runs either on a controller, which starts each test with the analyzer's `test` command, or on the analyzer itself, which runs the analyzer program in its `program` attribute [OAP-DC Section 10.3]. An analyzer's `runOn` member says which (Table 3).
2. The same rules apply to both (Section 14.2).
3. Whichever runs the schedule, the controller screens, corrects, and records the results (Sections 10 to 12).
4. Testing functions run in the controller's control process, as Automation Section 17.1.3 requires of every function.

7.2 The Testing Function

1. A testing function is a function of the type `water-testing`, which the Automation specification hosts [OAP-AU Section 11.10].
2. A testing function has the members of Table 2, each of its analyzers the members of Table 3, and each of its parameters the members of Table 4.

Table 2 — Members of a testing 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 whose water is tested.
vessel	identifier	OPTIONAL	The vessel samples are taken from. It MUST belong to system.
outputs	array	REQUIRED with analyzers	The output of every analyzer.
analyzers	array	OPTIONAL	The analyzers (Section 7.3).
parameters	array	OPTIONAL	The parameters whose results the function checks (Section 7.4).
extensions	object	OPTIONAL	Extensions, as in the Automation specification [OAP-AU Section 7.7].

3. A testing function MUST have `analyzers`, `parameters`, or both. One without analyzers drives no outputs: it checks the results that people enter and import, and reminds them to test.
4. `outputs` MUST list the `output` of every analyzer, and nothing else, so that a controller that does not support testing functions can still find their outputs, and treat them as Automation Section 19.2 requires for an unsupported function.
5. A testing function defines its own behavior when an input is unavailable (Section 16.2), so it MUST NOT have `onUnavailable`.
6. A configuration document that contains testing functions MUST be valid against the Automation schema, and each testing function MUST be valid against `$defs/testing` in `schema/water-testing.schema.json` [JSON Schema].
7. A configuration MAY have several testing functions, for example one for each system.
8. Revision 0.2 named the function type `org.openaquariumproject.water-testing`, and kept the saved state of every testing function under `state.extensions["org.openaquariumproject.water-testing"]`, keyed by function identifier. A controller SHOULD read a configuration that uses these names as if it used those of this revision, and SHOULD write the names of this revision.

The `analyzer` class is enabled while it runs, and is stopped with `set enabled false` [OAP-AU Table 5]. A testing function's demand on each of its analyzers is run, except while a `delayWhile` condition holds, testing is suspended outside a test on demand, or `program-not-loaded` has lasted its delay, when it is stop (Sections 8.5, 8.8, 10.5, and 14.3).

7.3 Analyzers

1. Each analyzer of a testing function is one output of the class `analyzer`, named in the function by its output identifier.
2. An analyzer MAY measure several parameters, in one test or in separate tests.

Table 3 — Members of an analyzer

Member	Type	Presence	Description
output	identifier	REQUIRED	The analyzer's output.
name, notes	string	OPTIONAL	Shown to the owner.
runOn	controller or analyzer	OPTIONAL	Where the schedule runs (Section 7.1). Default <code>controller</code> .
method	string	OPTIONAL	The Data Model method type of the analyzer's results, such as <code>titration</code> or <code>photometer</code> [OAP-DM Section 9.7]. Default: for each parameter, the method type that the analyzer's <code>methods</code> attribute gives it [OAP-DC Section 10.3], or else <code>other</code> .
schedule	array	OPTIONAL	The schedule entries (Section 8.2). Without it, the analyzer tests only on demand.
maxDelay	number (s)	OPTIONAL	How late a delayed test may start (Section 8.6). Default 3600 (provisional). At most 86 400.

Member	Type	Presence	Description
<code>delayWhile</code>	array	OPTIONAL	References to conditions while which tests wait (Section 8.5).
<code>maxRetests</code>	integer	OPTIONAL	The most retests after one scheduled test (Section 10.4). Default 2 (provisional).
<code>retestDelay</code>	number (s)	OPTIONAL	How long after a result a retest starts (Section 10.4). Default 300 (provisional).
<code>maxFailures</code>	integer	OPTIONAL	How many failures in a row suspend testing (Section 10.5). Default 3 (provisional).
<code>confirmFirst</code>	boolean	OPTIONAL	Whether the first result after a fresh start is held back until a second agrees (Section 10.3). Default <code>true</code> (provisional).
<code>reagents</code>	array	OPTIONAL	Reagent tracking (Section 9.1).
<code>waste</code>	object	OPTIONAL	Waste tracking (Section 9.5).
<code>corrections</code>	array	OPTIONAL	The corrections applied to the analyzer's results, one for each parameter at most (Section 11.5).
<code>compareDays</code>	number (days)	OPTIONAL	How often the analyzer should be compared with a reference (Section 11.7).
<code>extensions</code>	object	OPTIONAL	Extensions.

For example, an analyzer that tests alkalinity every 4 hours, and calcium and magnesium once a day:

```
{ "output": "analyzer", "method": "titration",
  "schedule": [{ "parameters": ["alkalinity"], "every": 14400 },
                { "parameters": ["calcium", "magnesium"], "times": ["02:15"] } ],
  "reagents": [{ "id": "alk", "shelfLifeDays": 30 }], "compareDays": 90 }
```

7.4 Parameters

A parameter entry holds what the owner says about the results of one parameter in one place, whatever their source: their plausible values, how old they may grow, how often to test by hand, and how large a correction may be.

Table 4 — Members of a parameter

Member	Type	Presence	Description
<code>parameter</code>	string	REQUIRED	A Data Model parameter identifier [OAP-DM Section 9.2].
<code>unit</code>	string	REQUIRED	The unit of every number of the entry. It MUST be accepted for <code>parameter</code> .
<code>plausible</code>	object	RECOMMENDED	Plausibility limits (Section 10.2): <code>min</code> , <code>max</code> , and <code>maxJump</code> .
<code>maxAge</code>	number (s)	OPTIONAL	How old an analyzer's latest accepted result may grow before <code>results-stale</code> is raised (Section 12.6). Default: see Section 12.6.
<code>remindDays</code>	number (days)	OPTIONAL	How long without an accepted result before <code>test-due</code> reminds the owner to test (Section 12.6).
<code>maxCorrection</code>	number	OPTIONAL	The most a correction may change a result before <code>correction-limit</code> is raised (Section 11.6). No default.
<code>notes</code>	string	OPTIONAL	Free text.

1. A result is checked against the entry of the testing function of its system whose `vessel` is the result's vessel, or, if there is none, against the entry of the testing function of its system that has no `vessel`. A result without a vessel describes the system's circulating water [OAP-DM Section 8.4].
2. A configuration MUST NOT have two entries for the same parameter in the same place, so that a result is judged by one set of limits.

7.5 Units and Times

1. The numbers of a parameter entry are in its `unit`. A controller converts each result to that unit with the Data Model registry before it screens it [OAP-DM Section 9.5].
2. Some numbers are differences rather than values: `maxJump`, `maxCorrection`, and the `offset` of a correction. A program that converts them to another unit MUST convert them with the factor alone, without the offset [OAP-AU Section 7.5].

3. A result's time is when its sample was taken: the analyzer result's `time` [OAP-DC Section 10.3], or the observation's `time` or its sample's `collected` time [OAP-DM Section 8.4].
4. 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. An analyzer program carries the time zone as `posixTimeZone` (Section 14.1).
5. Durations, such as `maxDelay` and `retestDelay`, are elapsed time.

7.6 Holds

1. An analyzer is held while its resolved state [OAP-AU Section 17.3] is stop: while an interlock, a mode, or an override holds it, while one of its `delayWhile` conditions is true or unknown (Section 8.5), while testing is suspended (Section 10.5), or while `program-not-loaded` has lasted its delay (Section 14.3.3).
2. It is also unable to test while its device is lost [OAP-FS Section 7.6], while it is under local control (`controlSource local`), and while it reports a fault that stops it, such as `hardware-failure`.
3. While an analyzer is held, no test starts. A test in progress MAY finish [OAP-FS Table 2], and its result is screened like any other: its sample was taken before the hold.
4. Tests that fall due while an analyzer is held, or unable to test, are delayed (Section 8.6).
5. The Automation presets stop analyzers in maintenance, water-change, and power-loss modes, and leave them to their function in feed and emergency modes [OAP-AU Table 29].
6. Water just after a water change has not yet mixed. Owners SHOULD extend the stop of a water-change mode on an analyzer by the time the new water takes to mix, such as 1800 s (provisional), with the mode's `extend` [OAP-AU Section 14.4]. One laboratory asks owners not to sample within 48 hours of a water change, but an analyzer tests the system as it is, which is what dosing needs.

7.7 Saved State

1. A controller MUST keep the state of each testing function across restarts, in its saved state [OAP-AU Section 17.8], as `state.functions["<function id>"]`: an object whose `analyzers` member is an object keyed by output identifier.
2. Its form is defined by `$defs/savedState` in `schema/water-testing.schema.json`, and Table 5 lists its members.
3. A controller MUST save it whenever it changes.
4. It is exported and imported with the configuration [OAP-AU Section 19], so that screening, suspension, and consumable estimates continue on a new controller.

Table 5 — Saved state of each analyzer

Member	Contents
<code>schedule</code>	For each schedule entry, in order, the latest test it has started, delayed, or skipped: <code>due</code> and <code>state</code> (<code>delayed</code> , <code>running</code> , <code>done</code> , or <code>skipped</code>), or null.
<code>results</code>	For each parameter: <code>accepted</code> , the last accepted exact result, and <code>suspect</code> , a result held back, each with <code>value</code> , <code>unit</code> , and <code>time</code> ; <code>retests</code> , the retests since the last scheduled test; <code>failures</code> , its failures in a row (Section 10.5); and <code>skipsOnly</code> , whether every one of those failures was a skipped test, so that recording a replaced reagent or an emptied waste container ends a suspension that they caused (Section 10.5.5).
<code>suspended</code>	When testing was suspended, while it is (Section 10.5).
<code>fresh</code>	Whether the next result follows a fresh start (Section 10.3).
<code>reagents</code>	For each reagent the controller tracks: <code>used</code> , the tests since it was replaced; <code>replaced</code> ; and <code>expires</code> .
<code>waste</code>	For waste the controller tracks: <code>used</code> , the tests since it was emptied; and <code>emptied</code> .
<code>compared</code>	For each parameter, when its last comparison was made (Section 11.7).

Revision 0.2 kept one count of failures in a row for the analyzer, as `failures`, although Section 10.5 counts them for each parameter. A controller that reads it counts it for each parameter that the analyzer tests on its schedule.

7.8 Faults and Alarms

1. A controller raises the faults of testing functions with the function as their subject [OAP-AN Section 8.2], naming the analyzer and the parameter. It raises `program-not-loaded` with the analyzer's output as its subject, as the Lighting Programs and Flow Programs specifications raise it for lights and pumps [OAP-LP].
2. Faults that the analyzer reports itself, such as `low` or `waste-full`, raise alarms with the analyzer's endpoint as their subject [OAP-AN Section 8.4]; the controller does not raise them again for the same condition.
3. Table 6 lists the codes water testing raises. Table 7 gives the codes this Specification adds, which are listed with their default severities in `conditions.json`.

Table 6 — Codes water testing raises

Code	Defined by	When	Latches
<code>low, empty</code>	Device Classes	A reagent is low or empty (Sections 9.2 and 9.3).	No
<code>expired</code>	Device Classes	A reagent is past its life (Section 9.4).	No
<code>waste-full</code>	Device Classes	The waste container is full (Section 9.5).	No
<code>implausible-result</code>	Device Classes	A scheduled test, with its retests, gave no accepted result (Section 10.5).	No
<code>calibration-due</code>	Device Classes	An analyzer has not been compared with a reference for <code>compareDays</code> (Section 11.7).	No
<code>input-unavailable</code>	Automation	A delay condition or a waste condition is unavailable (Section 16.2).	No
<code>command-refused</code>	Automation	An analyzer refused a command, such as <code>test</code> (Section 8.5).	No
<code>program-not-loaded</code>	Lighting Programs	An analyzer that should run its own schedule does not report the analyzer program its controller wrote (Section 14.3).	No
<code>results-stale</code>	This Specification	An analyzer's results are older than their maximum age (Section 12.6).	No
<code>testing-suspended</code>	This Specification	Testing stopped after repeated failures (Section 10.5).	Yes
<code>correction-limit</code>	This Specification	A correction changes results by more than the owner allows (Section 11.6).	No
<code>waste-nearly-full</code>	This Specification	The waste container is nearly full (Section 9.5).	No
<code>test-due</code>	This Specification	No accepted result of a parameter for <code>remindDays</code> (Section 12.6).	No

Table 7 — Conditions this Specification adds

Code	Meaning	Default	Critical for	Notes
<code>results-stale</code>	No accepted result of a parameter that an analyzer tests on its schedule is younger than the parameter's maximum age (Section 12.6).	<code>warning</code>	—	Its tests are failing, held, or skipped. Functions whose inputs read the results apply their unavailable-input behavior when each input's own maximum age passes; measurement-driven dosing then falls back, by default to its normal amount.
<code>testing-suspended</code>	A testing function stopped an analyzer's scheduled tests, because several in a row gave no accepted result (Section 10.5). It stays until a person clears it, or a test on demand gives an accepted result, or, when only skipped tests caused it, a replaced reagent or an emptied waste container is recorded.	<code>warning</code>	—	The analyzer may be out of reagent, drawing air, or faulty, or the water may really have changed: test by hand. Its results go stale, and dosing that uses them falls back.
<code>correction-limit</code>	A correction of an analyzer's results, proposed or applied, changes them by more than the owner's limit (Section 11.6).	<code>warning</code>	—	The analyzer or the reference result is wrong. Check both before relying on the correction.
<code>waste-nearly-full</code>	An analyzer's waste container is nearly full (Section 9.5).	<code>info</code>	—	Testing stops when it is full.

Code	Meaning	Default	Critical for	Notes
test-due	No result of a parameter has been accepted for longer than the owner's reminder interval (Section 12.6).	info	—	A reminder to test by hand. It clears when a result is accepted.

7.9 Visibility

A controller **MUST** show the owner, for each analyzer:

1. its next scheduled tests, and why a test is delayed, such as “held by water-change mode ‘Water change’, 25 min left” or “waiting for condition ‘After alkalinity dose’”;
2. its latest results, each with its status, its raw value, and its correction;
3. any suspect result, and the retest it waits for;
4. its reagents and waste, with their forecasts;
5. its last comparison and correction for each parameter, and when the next comparison is due; and
6. whether it runs its own schedule, and whether testing is suspended.

It **MUST** also show, for each parameter of a testing function, the latest accepted result from any source, with its source and age, and every unconfirmed result, waiting for a person to confirm its basis (Section 12.7).

8 Test Schedules

8.1 Overview

Analyzers test some parameters often and others rarely: one tests alkalinity up to 24 times a day and calcium and magnesium half as often, another tests nitrate and phosphate once a day, and some test at fixed times on chosen days. A schedule lists entries, each saying which parameters to test, or which maintenance action to run, and when.

8.2 Schedule Entries

Table 8 — Members of a schedule entry

Member	Type	Presence	Description
action	test, clean, or calibrate	OPTIONAL	What the analyzer does: a test, or its <code>clean</code> or <code>calibrate</code> command [OAP-DC Section 10.3]. Default <code>test</code> .
parameters	array	REQUIRED for test	The Data Model parameters the test measures. Not with other actions.
every	number (s)	OPTIONAL	The time between tests across the window, from <code>start</code> . Not with <code>times</code> .
start	time of day	OPTIONAL	When the window starts, and the first test falls due. Default <code>00:00</code> . Not with <code>times</code> .
end	time of day	OPTIONAL	When the window ends. Default: 24 hours after <code>start</code> . Not with <code>times</code> .
times	array of times of day	OPTIONAL	The times the entry falls due, instead of <code>every</code> , <code>start</code> , and <code>end</code> .
days	array of days	OPTIONAL	The days of the week that are testing days. Default: every day.
everyDays, from	integer, date	OPTIONAL	Testing days are <code>from</code> and every <code>everyDaysth</code> day after it. Both or neither.

1. Each parameter of a `test` entry **MUST** be one that the analyzer measures, as its `parameters` attribute lists [OAP-DC Section 10.3].
2. A schedule has at most 16 entries.
3. Without `every` and `times`, an entry falls due once on each testing day, at `start`.

For example, an analyzer that tests nitrate and phosphate at 23:00 every day, and cleans itself at 11:00 every second Sunday:

```
[{ "parameters": ["nitrate", "phosphate"], "start": "23:00" },
  { "action": "clean", "start": "11:00", "everyDays": 14, "from": "2026-10-04" }]
```

8.3 Testing Days

1. A local date is a testing day of an entry unless `days` or `everyDays` excludes it.
2. With `days`, only those days of the week are testing days.
3. With `everyDays` `N` and `from`, the date `from` and every `N`th date after it are testing days, and dates before `from` are not.
4. With both, a testing day is a date that both select.

These are the date filters of an Automation schedule [OAP-AU Section 10.6.1], and of a dosing program [OAP-DO Section 9.3].

8.4 Test Times

Whatever runs a schedule, a controller or an analyzer, MUST make each entry fall due as this section says.

1. With `times`, an entry falls due at those local times on each testing day, in order of time.
2. Otherwise, the window starts at `start` on the testing day, and ends at the first moment after its start with the local time `end`, or, without `end`, with the local time `start`, as a dosing window does [OAP-DO Section 9.4.2]. 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. With `every` `E`, the entry falls due at the window's start plus $k \times E$ seconds, for each whole number `k` from 0 for which $k \times E$ is less than `L`. Without `every`, it falls due once, at the window's start.
3. The tests of a testing day are those whose window starts on that date, even where the window ends the next day.

For example:

- `every` 14400 from `00:00` falls due at 00:00, 04:00, 08:00, 12:00, 16:00, and 20:00.
- `every` 18000 from `00:00` falls due at 00:00, 05:00, 10:00, 15:00, and 20:00, and again at 00:00 the next day.
- `every` 7200 from `22:00` to `06:00` falls due at 22:00, 00:00, 02:00, and 04:00, all on the testing day that starts at 22:00.
- On the day daylight saving time starts in Toronto, `every` 14400 from `00:00` falls due at 00:00, 05:00, 09:00, 13:00, 17:00, and 21:00 local time, every 4 hours of elapsed time, and `every` 3600 falls due 23 times, because the window to 00:00 the next day lasts 23 hours.

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

8.5 Starting a Test

A test starts when it falls due, or later while it is delayed, and it MUST NOT start unless all of these hold:

1. the clock is valid [OAP-FS Section 10.5];
2. the analyzer is not held, and is able to test (Section 7.6);
3. the analyzer is idle: its `state` is `idle` [OAP-DC Section 10.3];
4. no condition of `delayWhile` is true or unknown;
5. no reagent the test uses is empty (Section 9.3);
6. the waste is not full (Section 9.5); and
7. testing is not suspended (Section 10.5), unless the test is a test on demand (Section 8.8.5).

Then:

1. A controller starts a test with one `test` command listing its parameters, and a `clean` or `calibrate` entry with that command.
2. When the tests of several entries are due at once, the controller MAY give them as one `test` command listing all their parameters; otherwise it starts them one after another, in the order of the entries.

3. A test that the analyzer refuses MUST raise `command-refused` [OAP-AU Section 17.4], and is delayed as Section 8.6 says; the controller MUST NOT ask again more often than every 60 s (provisional).
4. `delayWhile` holds the analyzer through its function's demand (Section 7.2), so it also delays the tests of an analyzer that runs its own schedule (Section 14.3).

`delayWhile` keeps tests away from moments when the water is not typical, such as just after a dose of a product that changes a tested parameter. For example, tests that wait while the alkalinity head doses, and for 15 minutes after:

```
{ "id": "after-alk-dose", "type": "timer", "kind": "off-delay", "input": "alk-dosing", "duration": 900 }
```

with an output input `alk-dosing` of the head, and `"delayWhile": ["after-alk-dose"]` in the analyzer.

8.6 Delays and Skipped Tests

1. A test that cannot start when it falls due is delayed.
2. A delayed test MUST start as soon as Section 8.5 allows, if that is no later than `maxDelay` after it fell due, and before the next test of the same entry falls due.
3. Otherwise it MUST be skipped: it never starts, and it is recorded (Section 15.2). A skipped test raises nothing by itself; it may count as a failure (Section 10.5), and results that stop arriving raise `results-stale` (Section 12.6).
4. A test of a later entry that falls due while an earlier one is delayed waits behind it, in the order they fell due.
5. `maxDelay` MUST NOT be more than 86 400 s.

`maxDelay` covers a feed mode, a short maintenance mode, a dose's quiet period, or a restart, while a test long after its time would fall close to the next.

8.7 Tests and Doses

1. Measurement-driven dosing plans each dosing day when its first dose falls due, from the latest accepted result [OAP-DO Section 12.5]. Owners SHOULD schedule a test to end shortly before then; one maker advises dosing 30 minutes after each test.
2. A dose near the sample intake distorts a test taken soon after it. Sample intakes SHOULD be away from dosing outlets, as makers advise, and tests MAY wait after doses with `delayWhile` (Section 8.5).
3. A controller SHOULD warn the owner when no scheduled test of a parameter falls due in the 4 hours (provisional) before a dosing day that uses its results is planned.

8.8 Tests on Demand and Tests of a Standard

1. A person MAY start a test with the `run-action` operation [OAP-SA Table 6]. It starts as Section 8.5 says.
2. Its results are screened like any other (Section 10.3).
3. It does not move the schedule: the next scheduled test falls due as before, as one maker also does.
4. A person MAY start a test of a standard, which a controller sends as `test` with `standard true` [OAP-DC Section 10.3], with the `run-action` operation. Its results are not results of the system's water: the controller shows them, and records them in their test (Section 15.2), and never screens, uses, compares, or records them as observations (Section 12.2.3).
5. A person MAY start a test on demand while testing is suspended, to check the analyzer (Section 10.5.5). The controller enables the analyzer for the test, and holds it again when the test ends, unless an accepted result has ended the suspension.

8.9 Safety Checks

Table 9 — Schedule checks

Check	Presence	Applies to	What it requires	Section
Test times	REQUIRED	Controllers, analyzers	Tests fall due at the times Section 8.4 gives, on the testing days of Section 8.3, the same way on a controller and on an analyzer.	8.4
Starting tests	REQUIRED	Controllers, analyzers	A test starts only while the clock is valid, the analyzer is not held and is idle, no delay condition is true or unknown, no reagent it uses is	8.5

Check	Presence	Applies to	What it requires	Section
			empty, the waste is not full, and testing is not suspended. A test that the analyzer refuses raises <code>command-refused</code> , and is asked for again no more than every 60 s.	
Delays and skipped tests	REQUIRED	Controllers, analyzers	A test that cannot start when it falls due starts as soon as nothing prevents it, if that is within <code>maxDelay</code> of its due time and before the next test of its entry falls due. Otherwise it is skipped, and recorded.	8.6

9 Consumables

9.1 Reagents

Analyzers commonly track their reagents by counting what they use from a volume the owner enters, not by measuring what is left, and owners report counts that run far from the truth.

1. An analyzer with the Device Classes `reagents` feature tracks its reagents itself: each has an `id`, a `remaining` share, and an `expires` date, and, from Device Classes revision 0.2, the `tests` a full container gives and the `parameters` whose tests use it [OAP-DC Section 10.3]. The controller reads them, and records replacements with the analyzer's `replaceReagent` command.
2. Otherwise, a reagent entry with `tests` lets the controller estimate what is left: the share remaining is 100 % less 100 % times the tests since the reagent was replaced, divided by `tests`. Every test that uses it counts: scheduled tests, retests, tests on demand, tests of a standard, and tests that failed. The controller counts tests as Section 14.3.8 says, and keeps the count in the saved state (Section 7.7).
3. A person MAY correct an estimate by giving the share that is left.

Table 10 — Members of a reagent

Member	Type	Presence	Description
<code>id</code>	string	REQUIRED	Identifies the reagent. For an analyzer with the <code>reagents</code> feature, the <code>id</code> it reports.
<code>name</code>	string	OPTIONAL	Shown to the owner.
<code>parameters</code>	array	REQUIRED with <code>tests</code>	The parameters whose tests use the reagent.
<code>tests</code>	integer	OPTIONAL	How many tests a full container gives. Not used for an analyzer that tracks its reagents itself.
<code>lowDays</code>	number (days)	OPTIONAL	How long before the reagent runs out <code>Low</code> is raised (Section 9.2). Default 14 (provisional).
<code>shelfLifeDays</code>	number (days)	OPTIONAL	How long the reagent lasts once opened (Section 9.4).

9.2 Forecasts and Warnings

1. The forecast of a reagent is the number of days until it runs out: the share remaining, divided by the larger of the share used each day in the last 7 days, and the share the next 7 days' scheduled tests will use each day, where its `tests` is known.
2. `Low` is raised while the forecast is less than `lowDays`, and clears when a replacement is recorded.
3. `lowDays` defaults to 14 (provisional): reagents usually come from the analyzer's maker, by post. One maker warns 5 days ahead.
4. An analyzer that raises `Low` itself is not raised against again by the controller (Section 7.8.2), but the controller still shows the forecast.

9.3 Empty Reagents

Analyzers that run out of reagent do not all fail safely by themselves: one maker says its analyzer then reads high, until a result above its range stops it.

1. A reagent is empty when its `remaining` is 0, when the controller's estimate is at or below 0, or while the analyzer reports `empty` without saying which reagent.

2. While a reagent is empty, a test that uses it MUST NOT start (Section 8.5), and `empty` MUST be raised. Its tests are delayed, and then skipped (Section 8.6).
3. A result of a test that started while a reagent it uses was empty, or during which the analyzer reported it empty, MUST be rejected (Section 10.3).
4. An analyzer SHOULD refuse to start a test without enough reagent for it, as one maker's does.
5. After a replacement that follows `empty`, a controller SHOULD propose that the person prime the analyzer's reagent line before its next test, if the analyzer's maker describes one.
6. Nobody knows that a reagent is empty when the analyzer lacks the `reagents` feature and no reagent entry with `tests` covers a parameter it tests. A controller MUST then warn the owner, when the configuration is made and for as long as it stays so, that an empty reagent of that parameter may go unnoticed, and SHOULD advise giving the reagents' `tests`. It warns, and does not refuse the analyzer's results, because the owner may watch the reagent by eye (Appendix D, issue 29).

9.4 Reagent Life

Reagents lose strength once opened, and one maker gives its reagents 30 or 60 days once opened, and the makers of test kits 3 to 6 months.

1. A reagent past its `expires` date, or replaced more than `shelfLifeDays` ago, raises `expired`, which clears when it is replaced.
2. Testing continues, and the results of a test that used an expired reagent are recorded as such (Section 15.1), shown, and used by alarms. An aged reagent usually biases results slowly; screening and dosing's own limits bound what that can do, and stopping would leave dosing without results.
3. From 30 days (provisional) after a reagent expired, the results of a test that used it MUST NOT adjust dosing: a dosing head that would plan from them returns to its normal amount [OAP-DO Section 12.7]. They are still recorded, marked, shown, and used by alarms, so that a real change in the water is not missed.
4. Owners SHOULD set `shelfLifeDays` from the maker's advice.

9.5 Waste

1. An analyzer with the Device Classes `waste` feature reports its `wasteLevel`, and raises `waste-full` itself; the controller records emptying with its `emptyWaste` command [OAP-DC Section 10.3]. Its own `wasteTests` sets how many tests fill its container, and `waste.tests` is not used for it.
2. Otherwise, `waste.tests` lets the controller estimate the level: 100 % times the tests since the waste was emptied, divided by `waste.tests`.
3. `waste.full` names a condition that is true when the waste container is full, such as a level switch near its top.
4. The waste is full while the analyzer reports `waste-full`, its `wasteLevel` or the controller's estimate is at or above 100 %, or `waste.full` is true or unknown. While it is full, a test MUST NOT start, and `waste-full` MUST be raised.
5. `waste-nearly-full` is raised while the level is at or above `waste.warnLevel`, which defaults to 80 % (provisional).

Table 11 — Members of `waste`

Member	Type	Presence	Description
<code>tests</code>	integer	OPTIONAL	How many tests fill an empty waste container. Not used for an analyzer that tracks its waste itself.
<code>full</code>	reference	OPTIONAL	A condition that is true when the waste is full.
<code>warnLevel</code>	number (%)	OPTIONAL	The level at which <code>waste-nearly-full</code> is raised. Default 80 (provisional).

Analyzer waste holds reagents, some of them acidic or harmful, and one maker requires its waste to be neutralized before it is poured away. Some makers send waste back to the tank, and say that a titration's waste only removes a little alkalinity; others never do.

6. An analyzer MUST NOT return to the system water that has met a reagent, whether it measures with it, checks itself with it, or cleans with it, unless its maker documents that doing so is harmless, with its effect on the water, and the owner chooses it. By default, it MUST send such water to waste.
7. An analyzer's maker SHOULD say how its waste is to be disposed of.
8. Nobody knows that the waste is full when the analyzer lacks the `waste` feature, and its configuration has neither `waste.tests` nor `waste.full`. A controller MUST then warn the owner, when the configuration is made and for as long as it stays so, that a full waste container may go unnoticed.

9.6 Replacing Reagents and Emptying Waste

1. A person records a reagent's replacement with the `record-maintenance` operation [OAP-SA Table 6], which the controller carries out with the analyzer's `replaceReagent` command, or, for a reagent the controller tracks, by itself. They MAY give its `expires` date and its lot.
2. Recording a replacement resets the estimate, clears `low`, `empty`, and `expired`, and marks a fresh start, so that the next result is confirmed (Section 10.3). It also ends a suspension that only skipped tests caused (Section 10.5.5).
3. Makers advise not to mix old and new reagent, and say that a new batch can shift results. A controller SHOULD propose a comparison with a reference after a reagent of a new lot is in use (Section 11.7).
4. A person records that the waste was emptied with the `record-maintenance` operation, which the controller carries out with the analyzer's `emptyWaste` command, or by itself. It resets the estimate, clears `waste-full` and `waste-nearly-full` unless their conditions still hold, and ends a suspension that only skipped tests caused (Section 10.5.5).
5. A controller SHOULD record each replacement and emptying (Section 15.4).

9.7 Safety Checks

Table 12 — Consumable checks

Check	Presence	Applies to	What it requires	Section
Reagent forecasts	RECOMMENDED	Controllers	Each reagent's remaining amount is read from the analyzer or estimated from its tests, and forecast to run out from the coming week's scheduled tests and the last week's use. <code>low</code> is raised when the forecast is less than <code>lowDays</code> .	9.2
Empty reagents	REQUIRED	Controllers, analyzers	No test that uses an empty reagent starts, and <code>empty</code> is raised. A result of a test that started while, or ran while, a reagent it uses was empty is rejected.	9.3
Consumables nobody tracks	REQUIRED	Controllers	A controller warns the owner, from the moment the configuration is made, about an analyzer whose empty reagents or full waste neither it nor the analyzer can detect.	9.3
Reagent life	RECOMMENDED	Controllers, analyzers	A reagent past its <code>expires</code> date, or replaced more than <code>shelfLifeDays</code> ago, raises <code>expired</code> , which clears when it is replaced. Testing continues, and its results record that the reagent had expired.	9.4
Expired reagents and dosing	REQUIRED	Controllers	From 30 days after a reagent expired, the results of tests that used it adjust no dosing, which returns to its normal amount. They are still recorded, marked, shown, and used by alarms.	9.4
Full waste	REQUIRED	Controllers, analyzers	While the waste container is full, or its <code>full</code> condition is true or unknown, no test starts, and <code>waste-full</code> is raised.	9.5
Waste nearly full	RECOMMENDED	Controllers	<code>waste-nearly-full</code> is raised while the waste container is at or above <code>warnLevel</code> .	9.5
No reagent into the system	REQUIRED	Analyzers	An analyzer sends water that has met a reagent, whether it measures with it, checks itself with it, or cleans with it, to waste. It returns such water to the system only when its maker documents that this is harmless, with its effect on the water, and the owner has chosen it.	9.5

10 Plausibility

10.1 Overview

Analyzers already retest results that look wrong: one retests any result more than 0.2 dKH from the last, others retest above an owner's threshold, and one never acts on its first result after a restart. None publishes the same rule, and a result can be judged again, differently, by the dosing that uses it. This section defines one rule, applied by the controller to every result, before anything uses it.

10.2 Plausibility Limits

`plausible` holds the owner's limits for a parameter, in its entry's `unit`:

1. `min` and `max` bound the plausible range: the values the parameter cannot leave in this system unless a measurement is wrong. They are not the owner's acceptable range, which raises alarms [OAP-AN Section 9], and SHOULD lie well outside it.
2. `maxJump` is the largest difference from the last accepted result of the same source that is believed without a second result.
3. They are the members of a dosing head's `adjust.plausible`, with the same meaning [OAP-DO Section 12.4].

Plausibility limits have no defaults: what values are impossible depends on the system and its owner. Table 25, published in `starting-points.json`, gives starting points for some kinds of system, by water type and system tag, from the measuring ranges of analyzers.

4. A controller SHOULD propose limits for each parameter that has none: Table 25's starting point for the system's water type and tags, where there is one, and otherwise limits from the analyzer's measuring range and the owner's ranges.
5. It proposes `maxJump` from the longest interval between the analyzer's scheduled tests of the parameter (Section 12.6): Table 25's smaller value for tests up to 4 hours (provisional) apart, and its larger value for tests further apart, or for results without an analyzer's schedule.
6. A controller MUST NOT apply proposed limits until the owner has confirmed them.

For example, a reef owner might set alkalinity 5 to 14 dKH with a `maxJump` of 0.5 dKH, and calcium 300 to 550 mg/L with a `maxJump` of 40 mg/L, where Table 25 proposes 4 to 16 dKH and 250 to 600 mg/L, with `maxJump` 0.2 dKH and 20 mg/L for tests every 4 hours.

10.3 Screening a Result

A controller MUST screen every result of the system's water before it is used, by these rules, in order:

1. **Reagent.** A result of a test that ran with an empty reagent (Section 9.3) is rejected.
2. **Range.** A result outside `min` and `max` is rejected. A result with a qualifier [OAP-DM Section 8.4] is rejected when it is certainly outside: `>` at or above `max`, or `<` at or below `min`. Otherwise a qualified result is accepted, as a limit: it settles no suspect, and is not the last accepted result for rule 4.
3. **Fresh start.** With `confirmFirst`, the first result of an analyzer after a fresh start is held back as suspect, unless it agrees with a suspect already held, that is, is within `maxJump` of it; then both are accepted.
4. **Jump.** A result within `maxJump` of the last accepted result of its source is accepted, and a suspect held is rejected. Otherwise, a result within `maxJump` of a suspect held is accepted, and so is the suspect, because two results agree. Otherwise, if its source has no accepted result and holds no suspect, it is accepted. Otherwise, it is held back as suspect, and a suspect held before is rejected.
5. Without `plausible`, rules 2 to 4 do not apply, and without `maxJump`, rules 3 and 4 do not apply: the result is accepted.

Then:

6. An accepted result is used, and recorded as a Data Model observation (Section 15.1). A suspect result that is later accepted is recorded then, with its own sample time.

- Suspect and rejected results are never used: no input reads them, they raise and clear no range alarm, and they are not observations. They are recorded with the test that produced them (Section 15.2), and shown to the owner with their values.
- The source of an analyzer's result is the analyzer. Manual and laboratory results are screened on entry instead (Sections 12.3 and 12.4).

Table 13 shows the rules at work for an analyzer with `min` 5, `max` 14, and `maxJump` 0.5, in dKH, whose last accepted result is 8.2 when the sequence starts.

Table 13 — Screening a sequence of results

Result	Held before	Status	Held after	Why
8.4	—	accepted	—	Within 0.5 of 8.2.
6.6	—	suspect	6.6	1.8 from 8.4.
8.1	6.6	accepted	—	Within 0.5 of 8.4; 6.6 is rejected.
6.6	—	suspect	6.6	1.5 from 8.1.
6.5	6.6	accepted	—	1.6 from 8.1, but within 0.5 of 6.6, which is accepted too: the alkalinity fell.
> 14	—	rejected	—	Certainly above <code>max</code> .

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

10.4 Retests

- When a result is held back or rejected, or a test fails without a result of the parameter, such as a test whose `testEnded` reports `failed` or `aborted`, the controller MUST ask the analyzer for a retest of that parameter, `retestDelay` after the result or the failure, unless item 2 forbids it.
- The controller MUST NOT ask for more than `maxRetests` retests after one scheduled test, whatever their results.
- A retest starts as Section 8.5 says, and waits for holds and `delayWhile`. If it has not started when the next scheduled test of the parameter falls due, it is dropped, and that test's result settles any suspect.
- A retest's result is screened like any other.
- Only the testing function that drives an analyzer sends it `test` commands. A dosing head that holds a result back [OAP-DO Section 12.4.6] does not send `test` itself; a retest it needs is given by the testing function, within these limits.
- Retests use reagent and fill the waste, and count in their forecasts (Sections 9.2 and 9.5).

`retestDelay` lets the analyzer flush its last sample and draw fresh water, while the water itself has barely changed.

10.5 Repeated Failures

- A scheduled test fails for a parameter when neither it nor any of its retests gives an accepted result of the parameter, once no further retest can follow it: its retests are used up, or the next scheduled test of the parameter has fallen due.
- A scheduled test of a parameter that is skipped (Section 8.6) is a failure, unless the analyzer was held or unable to test (Section 7.6), or the clock was not valid, for the whole time it was delayed. So a skip because a reagent is empty, the waste is full, or the analyzer refused the test counts, and a long hold, such as a maintenance mode, does not; a suspension that only skips caused ends when their cause is fixed and recorded (item 5).
- A failure raises `implausible-result`, naming the analyzer, the parameter, and the results that were not accepted. It clears when a result of the analyzer and parameter is accepted.
- After `maxFailures` failures in a row for any one parameter, the controller MUST suspend the analyzer's testing: no scheduled test or retest starts, its function's demand on it is stop (Section 7.2), and `testing-suspended` is raised.
- `testing-suspended` latches, and is kept in the saved state across restarts. The controller MUST end the suspension, and resume testing, when one of these happens, and at no other time:

- when a person who has checked the analyzer clears it, with the `clear-latch` operation [OAP-SA Table 6];
 - when a test on demand that a person starts gives an accepted result of each parameter whose failures suspended testing (Section 8.8.5); or
 - when only skipped tests caused it, and a person records a replaced reagent or an emptied waste container (Section 9.6).
6. A suspension that a person clears, or that a replaced reagent or an emptied waste container ends, is a fresh start, so the analyzer's next result is confirmed before it is used (Section 10.3). A test on demand that ends it has given an accepted result already.
 7. An accepted result resets the count of failures in a row for its parameter.

An analyzer that keeps failing wastes reagent, fills its waste, and may be drawing air or leaking. One maker's analyzer stops testing after two very low results in a row, more than four retests, or a result above its range, until it is restarted or cleared. The results go stale, and dosing that uses them falls back as its `fallback` says, by default to its normal amounts [OAP-DO Section 12.7]. A real change in the water can look like a fault, so the alarm names the values: an owner who sees them SHOULD test by hand. An empty reagent or a full waste container only delays tests until they are skipped, so a suspension that only skipped tests caused ends when the owner fixes the cause and records it.

10.6 Results Judged Once

The Dosing specification asks that a result not be judged by two sets of limits [OAP-DO Appendix D, issue 13].

1. Section 10.3 uses the rule of Dosing Section 12.4, with the same members, adding qualifiers, fresh starts, and empty reagents.
2. A dosing head whose input reads results that a testing function screens (Section 13.1) SHOULD leave out `adjust.plausible`: its results have already passed. With it, a result that a retest has confirmed is held back again by the head, until the next scheduled test.
3. A controller SHOULD warn the owner about a head that has both, and SHOULD say if their limits differ.

10.7 Safety Checks

Table 14 — Plausibility checks

Check	Presence	Applies to	What it requires	Section
Screening results	REQUIRED	Controllers	Every result is screened before it is used. A result outside the plausible range, or from a test that ran with an empty reagent, is rejected. A result that differs from the last accepted result by more than <code>maxJump</code> , or the first after a fresh start, is held until a later result agrees. Only accepted results are used.	10.3
Retests	REQUIRED	Controllers	A rejected or held result, or a failed test, leads to a retest <code>retestDelay</code> later, at most <code>maxRetests</code> times for each scheduled test.	10.4
Repeated failures	REQUIRED	Controllers	After <code>maxFailures</code> scheduled tests of a parameter in a row give no accepted result of it, the analyzer's scheduled tests stop, and <code>testing-suspended</code> is raised. It stays, across restarts, until a person clears it, a test on demand gives an accepted result, or, when only skipped tests caused it, a replaced reagent or an emptied waste container is recorded.	10.5

11 Corrections

11.1 Overview

Every analyzer has a bias: its results differ from the true value by a roughly constant amount or share, through its reagent's strength, its pumps, its detector, or its installation [VIM 2.18]. Makers let owners correct for it: some multiply every result by a factor, one adds an offset, limited to ± 2 dKH, and some take a reference value and do not say which. Laboratories themselves differ: five laboratories given the same water reported calcium from 414 to 433 mg/L.

This section defines how an analyzer is compared with reference results, and how a correction is proposed, confirmed, applied, and recorded. A correction does not repair an analyzer: makers advise calibrating its pumps and sensors first, and correcting only a difference that persists.

11.2 Reference Results

1. A reference result is a result of the same water from a method the owner trusts more than the analyzer: a laboratory result, the owner's own careful test, or a reference solution of known value.
2. A person chooses which results are references. A controller SHOULD propose each laboratory result of a parameter an analyzer measures as one.
3. A reference SHOULD be the mean of two or more tests where the method allows, because a single test kit result can be off by 0.3 dKH or more.
4. A reference MUST be of the same Data Model parameter as the analyzer's result. A laboratory's phosphorus by ICP is `element:P`, not `phosphate`, and is not a reference for a phosphate analyzer [OAP-DM Section 9.3].
5. Parameters that change in a sample while it is posted, such as nitrate, SHOULD NOT be used as references unless the laboratory preserves them.
6. An unconfirmed result is never a reference (Section 12.7).

11.3 Comparisons

1. A comparison pairs a reference result with the analyzer's accepted result of the same parameter and place whose sample was taken nearest in time to the reference's, within the comparison window of 3600 s (provisional).
2. It uses the analyzer's raw result, never a corrected one, so that a correction is never built on another.
3. Its difference is the reference less the raw result, and its ratio the reference divided by the raw result, both in the parameter entry's unit.
4. When no result of the analyzer is within the window, there is no comparison. A controller SHOULD offer to start a test on demand when a person records a reference result, and SHOULD advise people to take reference samples at the moment the analyzer samples.
5. A controller SHOULD record each comparison (Section 15.3), and keep its time as `compared` in the saved state.

11.4 Proposing a Correction

1. A controller proposes a correction from the latest three comparisons (provisional) of the parameter within `compareDays`, or within 90 days without it, and only once it has three, so that one wrong reference cannot set it alone.
2. With the method `factor`, the default, the correction's factor is the sum of their references divided by the sum of their raw results, and its offset is 0.
3. With the method `offset`, its offset is the mean of their differences, and its factor is 1.
4. A person MAY choose other comparisons, fewer of them, the method, or a factor and offset of their own. A controller SHOULD then say that a correction from fewer than three comparisons rests on fewer references.
5. A controller SHOULD show the spread of the comparisons, and SHOULD advise against a correction smaller than that spread, or smaller than the reference method's uncertainty, as makers advise correcting only differences above 0.2 to 0.3 dKH, or 5 %.
6. A proposal stands until a person applies a correction of the parameter or dismisses the proposal, and a comparison made after that makes a new one.

For example, laboratory results of 438, 446, and 452 mg/L of calcium, against raw analyzer results of 420, 428, and 433 mg/L, give a factor of $1336 \div 1281 = 1.043$, or an offset of +18.3 mg/L.

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

11.5 Applying a Correction

1. A controller MUST NOT apply a correction until a person confirms it, with the `calibrate` operation [OAP-SA Table 6]. No result, reference, or import changes a correction by itself, because a wrong reference, or a sample mixed up in the post, would then change dosing.
2. Before the person confirms it, the controller MUST show the comparisons it comes from, the previous correction, and how much it changes the latest result.
3. A corrected result is the raw result times `factor`, plus `offset`, the order of the Device Classes `corrections` attribute [OAP-DC Section 10.3].
4. A correction MUST be applied exactly once:
 - for an analyzer with the `correction` feature, the controller writes the corrections to its `corrections` attribute within a minute of their differing, and the analyzer applies them to every result it reports; the controller does not apply them again;
 - for any other analyzer, the controller applies them to each raw result before it screens it.
5. The controller writes the offset of each correction in the correction's own `unit`, with that `unit`, to an analyzer of Device Classes revision 0.2 or later, and converts it to the parameter's canonical unit as a difference for an analyzer of revision 0.1, whose `corrections` have no `unit` [OAP-DC Section 10.3].
6. A correction applies to results whose samples are taken after it is applied. Results recorded before it keep their values.
7. The raw result of an analyzer that corrects its own results is the `raw` it reports with each result, and, without `raw`, its reported value less the offset, divided by the factor, of the corrections it held.
8. Corrections are part of the configuration: each analyzer's `corrections` (Table 15), exported with it.

Table 15 — Members of a correction

Member	Type	Presence	Description
<code>parameter</code>	string	REQUIRED	The Data Model parameter corrected. One correction for each parameter at most.
<code>factor</code>	number	OPTIONAL	Multiplies the raw result. Greater than 0. Default 1.
<code>offset</code>	number	OPTIONAL	Added after the factor, in <code>unit</code> . Default 0.
<code>unit</code>	string	REQUIRED with <code>offset</code>	The unit of <code>offset</code> . MUST be accepted for <code>parameter</code> .
<code>set</code>	timestamp	OPTIONAL	When the correction was applied.
<code>notes</code>	string	OPTIONAL	Free text, such as what it was based on.

A correction has `factor`, `offset`, or both.

11.6 Limits on Corrections

1. `maxCorrection` is the most a correction may change a result, in the parameter entry's unit: the difference between a raw result and the same result corrected.
2. While the correction proposed from the latest comparisons, as long as the proposal stands (Section 11.4.6), or the correction applied, changes the analyzer's latest accepted raw result by more than `maxCorrection`, `correction-limit` is raised, naming the analyzer, the parameter, and the change.
3. A large correction means the analyzer, or the reference, is wrong. A controller SHOULD advise the person to check both, by calibrating the analyzer and testing the reference again, before applying it.
4. The person MAY still apply it; `correction-limit` then stays while it is applied.
5. For a parameter whose entry has no `maxCorrection`, a controller MUST NOT apply a correction that changes the analyzer's latest accepted raw result by more than 10 % (provisional), until the person has confirmed that size, shown to them.
6. A controller SHOULD propose `maxCorrection` for each parameter that has none, from Table 25's starting point for the system, where there is one, and MUST NOT apply it until the owner has confirmed it.

`maxCorrection` has no default. Table 25's are three times the combined uncertainty of an analyzer and a laboratory, the square root of the sum of the squares of their makers' stated accuracies. One maker limits its offset to ± 2 dKH; another tells owners to stop using a test kit that reads more than 50 to 75 % from its standard.

11.7 When to Compare

1. With `compareDays`, an analyzer whose results of a parameter it tests on its schedule have not been compared with a reference for that many days raises `calibration-due`, which clears at the next comparison. Testing continues.
2. A controller SHOULD propose `compareDays` of 90 (provisional): hobby laboratories suggest testing every 2 to 3 months.
3. Owners SHOULD also compare after a reagent of a new lot is in use, and after the analyzer is serviced.

11.8 The Analyzer's Own Calibration

Many analyzers calibrate their own pumps and sensors, against a standard solution or buffers, by themselves or with the `calibrate` command.

1. A schedule entry with `action calibrate` runs the `calibrate` command.
2. `calibration-failed` and `calibration-due` that the analyzer reports raise alarms [OAP-AN Section 8.4].
3. The analyzer's own calibration is a fresh start (Section 10.3).
4. It does not replace comparison with a reference, and it does not change the corrections; owners SHOULD compare again after it.
5. An analyzer does not report what its own calibration computed, so its record holds no result (Section 15.3.5).

11.9 Safety Checks

Table 16 — Correction checks

Check	Presence	Applies to	What it requires	Section
Corrections need a person	REQUIRED	Controllers	A correction is applied only after a person confirms it with the <code>calibrate</code> operation. No result changes a correction by itself.	11.5
Corrections applied once	REQUIRED	Controllers, analyzers	A correction is applied exactly once: by an analyzer with the <code>correction</code> feature, to which the controller writes it, or by the controller. A corrected result is the raw result times <code>factor</code> , plus <code>offset</code> .	11.5
Comparisons	RECOMMENDED	Controllers	Each reference result is compared with the analyzer's accepted raw result of the same parameter and place whose sample was taken nearest in time, within the comparison window, and the comparison is recorded.	11.3
Correction limit	RECOMMENDED	Controllers	While a correction proposed from the latest comparisons, or the correction applied, changes results by more than <code>maxCorrection</code> , <code>correction-limit</code> is raised.	11.6
Comparison due	RECOMMENDED	Controllers	With <code>compareDays</code> , an analyzer whose results of a parameter have not been compared with a reference for that many days raises <code>calibration-due</code> , which clears at the next comparison. Testing continues.	11.7

12 Results

12.1 Sources

Table 17 — Where results come from

Source	How it arrives	How it is checked	Its time
An analyzer that a testing function drives	The analyzer's <code>result</code> events and <code>lastResults</code>	Corrected, then screened, with retests (Sections 10 and 11)	The sample time the analyzer reports
A manual result	A person enters it	Checked on entry, with the person's confirmation (Section 12.3)	When the person took the sample
A laboratory result	A person imports a Data Model document	Checked on import, with the person's confirmation (Section 12.4)	The sample's <code>collected</code> time

Source	How it arrives	How it is checked	Its time
An analyzer that no testing function drives	As Automation Section 9.2 reads it	Not by this Specification; dosing checks it [OAP-DO Section 12.4]	The sample time the analyzer reports

Every analyzer SHOULD be driven by a testing function, so that its results are screened.

12.2 Analyzer Results

1. A controller MUST take each result of an analyzer from its `result` event, or, if it missed the event, for example during a restart or a loss of contact, from `lastResults` when it next reads it. Over the Device Protocol, a controller that subscribes with `eventsAfter` also receives the `result` events it missed, as far back as the device keeps them [OAP-DP Section 11.6]. On the module bus, a controller asks a module for the events it kept while the controller was not listening, when contact returns [OAP-MB Section 11.4].
2. A result is new when its time is later than that of the last result of the same parameter the controller took from the analyzer; a controller MUST take a result once.
3. A result with `standard true` is of a standard, not of the system's water [OAP-DC Section 10.3]. The controller shows it, and records it in its test (Section 15.2), and MUST NOT screen it, use it, compare it, or record it as an observation.
4. The controller MUST treat a result as having no valid time while the analyzer reports `clock-invalid` [OAP-DC Section 7.6], while it reports its time as unknown, and when its time is more than 60 s (provisional) later than the controller's own clock. It then uses the time it received the result as the sample time [OAP-FS Section 10.2.2], and takes the result only from a `result` event, or from `lastResults` once after a `testEnded` event whose `result` events it missed, never from a later reading of `lastResults`, which would give an old result a new time.
5. The controller corrects each new result (Section 11.5), screens it (Section 10.3), and records it (Section 15).

12.3 Manual Results

1. A person enters a manual result, with the `enter-results` operation [OAP-SA Table 6], giving its parameter, value, unit, the method and instrument, the place, and the time the sample was taken, which defaults to the time of entry. A controller MUST NOT record a result entered by a person who is not allowed `enter-results`, because results steer dosing (Section 20, item 2).
2. The unit MUST be accepted for the parameter [OAP-DM Section 9.2]. The controller SHOULD show the value converted to the parameter entry's unit before it is recorded, so that a result as N is not taken for one as NO₃.
3. When the instrument's documentation does not state the result's chemical basis, the controller asks the person which basis it is, showing what each would mean, as Data Model Section 9.4 says. A result whose basis the person cannot give is unconfirmed (Section 12.7).
4. Before recording it, the controller checks it against the parameter's `plausible`: its range, and `maxJump` from the latest accepted result of the parameter from any source. A result that fails MUST be recorded only when the person confirms it, and is discarded otherwise.
5. A confirmed result is accepted, and recorded as confirmed by a person (Section 15.1).
6. Whether to test again is the person's choice.

Test kits differ: a retailer's comparison found alkalinity kits up to 1.85 dKH apart on the same water. The check on entry is not a verdict on the kit; it catches a misplaced decimal point or a wrong unit.

12.4 Laboratory Results

1. Laboratory results arrive as Data Model documents, with Samples that name the `laboratory`, and Observations of them [OAP-DM Section 8.5]. A person imports them with the `enter-results` operation, and a controller MUST NOT import them for a person who is not allowed it. The controller asks the person which system they belong to, as Data Model Section 8.5 says.
2. Each result of a parameter that has an entry is checked against its plausible range. A result outside it is recorded only when the person confirms it.

3. A laboratory result's time is its sample's `collected` time, days or more than a week before it arrives. It is the latest result of its parameter only if no later result exists (Section 12.5).
4. Results of parameters without an entry, such as most `element` parameters, are recorded as they are.
5. A controller MUST NOT import the same laboratory sample twice; it recognizes it by its identifier, or by its `laboratory.reference` [OAP-DM Section 7.3], and tells the person that it has the sample already.
6. A controller SHOULD propose each laboratory result of a parameter an analyzer measures as a reference (Section 11.2).
7. A laboratory result whose `unitAssumed` is `true` is unconfirmed (Section 12.7).

Laboratories deliver results as web pages, PDF files, CSV files, and, for one, a documented interface. How they are turned into Data Model documents is outside this Specification.

12.5 The Latest Result

1. The latest result of a parameter in a place is the accepted result with the latest sample time, whatever its source, among the results for that place [OAP-AN Section 9.3.1].
2. A result with an older sample time than the latest does not replace it, wherever it came from.
3. A result whose sample time is later than the controller's clock is never the latest (Section 12.7).

12.6 Stale and Missing Results

1. For each parameter that an analyzer tests on its schedule, a controller MUST raise `results-stale` while the analyzer's latest accepted result of it is older than the parameter's `maxAge`, naming the analyzer and the parameter. It clears when a result is accepted.
2. `maxAge` defaults to twice the longest interval between the analyzer's scheduled tests of the parameter, plus its `maxDelay` (provisional), so that one failure does not make results stale, and two do. For tests every 4 hours with the default `maxDelay`, it is 9 hours.
3. An input that reads the results becomes unavailable when its own `maxAge` passes [OAP-AU Section 9.8], and the functions that use it apply their own behavior; measurement-driven dosing falls back as its `fallback` says, by default to its normal amount [OAP-DO Section 12.7]. An input whose `maxAge` is the parameter's, as Section 13.1.4 advises, becomes unavailable as `results-stale` is raised; the default `maxAge` of an analyzer's input, 48 hours [OAP-AU Table 10], is far longer. `results-stale` tells the owner even when no function uses the results.
4. With `remindDays`, `test-due` is raised while no accepted result of the parameter, from any source, has a sample time within that many days. It clears when a result of the parameter is accepted.
5. Ages are measured as Fail-Safe Behavior Section 10.2.2 says.

Analyzers report a missing test in their own ways, such as an error after 24 hours without a test, or a stale-sample notice after 1.5 times the test interval.

12.7 Assumed Bases and Times Later Than the Clock

A result whose chemical basis was only assumed can be wrong by up to 4.4 times, and a result whose sample time is later than the controller's clock would stay the latest result for ever, never age, and hide every real result after it [OAP-DM Sections 7.4 and 9.4].

1. An unconfirmed result is one whose chemical basis was only assumed: an entered result whose basis the person could not give, recorded in the unit Data Model Section 9.4 assumes, and a laboratory result whose `unitAssumed` is `true`.
2. A controller MUST NOT use an unconfirmed result: no input reads it, it raises and clears no range alarm, it is not a reference or part of a comparison, and it is not the latest result. The controller records it as an observation with `unitAssumed true` [OAP-DM Section 9.4], shows it to the owner as waiting for a basis, and asks a person to confirm one.
3. A person confirms a basis by choosing it. The controller then records the observation in the unit of the basis chosen, without `unitAssumed`, as Data Model Section 9.4 says, and screens and uses it as an entered result (Section 12.3).
4. A controller MUST NOT record an entered or imported result whose sample time is later than its own clock, which a valid clock shows to be impossible [OAP-DM Section 7.4]. It shows the person the time, and

records the result once the person has corrected it, and a document's results once the person has corrected the document or the controller's clock.

- While the controller's clock is not valid [OAP-FS Section 10.5], it records entered and imported results as they come, and checks their times when its clock is valid again. A result that then proves later than the clock is treated as item 4 says: from then on, nothing uses it, the controller shows the person its time, and it records the result again, in place of the first record, once the person has corrected it.

12.8 Safety Checks

Table 18 — Result checks

Check	Presence	Applies to	What it requires	Section
Analyzer results	REQUIRED	Controllers	Every result of an analyzer is taken once, from its <code>result</code> event or from <code>lastResults</code> , with the time its sample was taken, and is corrected, screened, and recorded. While the analyzer's time is not valid, results are taken only from events or once after a test ends. A result of a standard is never an observation.	12.2
Entered results	REQUIRED	Controllers	A manual or laboratory result outside the plausible range, or a manual result that differs from the latest accepted result by more than <code>maxJump</code> , is recorded only after a person confirms it. The controller shows each entered result in the parameter's unit.	12.3
Laboratory samples imported once	REQUIRED	Controllers	A controller never imports the same laboratory sample twice, whether it recognizes it by its identifier or by its <code>Laboratory.reference</code> .	12.4
Stale results	REQUIRED	Controllers	While an analyzer's latest accepted result of a parameter it tests on its schedule is older than the parameter's <code>maxAge</code> , <code>results-stale</code> is raised.	12.6
Reminders to test	OPTIONAL	Controllers	With <code>remindDays</code> , <code>test-due</code> is raised while no accepted result of the parameter, from any source, is younger than that many days, and clears when one is accepted.	12.6
Results with an assumed basis	REQUIRED	Controllers	A result whose chemical basis was only assumed is recorded with <code>unitAssumed</code> , and no input, range alarm, comparison, or correction uses it until a person confirms its basis.	12.7
Sample times later than the clock	REQUIRED	Controllers	A controller records no entered or imported result whose sample time is later than its clock, and never takes one as the latest result.	12.7

13 Use by Other Functions

13.1 Inputs

- An Automation `endpoint` input of the class `analyzer` bound to the same endpoint as an analyzer of a testing function MUST read that analyzer's latest accepted result of its parameter, corrected, with its sample time as the reading's time, rather than `lastResults` as it is [OAP-AU Section 9.2].
- An `observation` input [OAP-AU Section 9.5] reads the latest accepted result from any source: suspect and rejected results are not observations (Section 10.3.7), and the input passes over unconfirmed results and results dated later than the clock, which the Data Model says not to act on (Section 12.7) [OAP-DM Sections 7.4 and 9.4]. With `methods`, it can choose between methods, but not between an analyzer and a person's test of the same method type, such as a titration [OAP-DM Table 25]; an `endpoint` input follows one analyzer.
- A qualified result is a limit, not a value [OAP-AU Section 9.2.2].
- An input of an analyzer's results SHOULD have a `maxAge` between 1.5 and 3 times the longest interval between the analyzer's scheduled tests of its parameter, as Dosing Section 12.3.3 says, and a controller SHOULD propose the parameter's `maxAge` (Section 12.6), so that the input and `results-stale` agree.

An `endpoint` input follows one analyzer closely, with a short `maxAge`; an `observation` input uses the latest of an analyzer, the owner's tests, and laboratory results, and suits results that come rarely.

13.2 Dosing

1. Dosing uses results through its inputs [OAP-DO Section 12.3], and so only accepted results.
2. A head whose input reads screened results leaves out its own plausibility limits (Section 10.6), and retests are given by the testing function (Section 10.4.5).
3. A new correction changes results, and so the next day's dosing. A controller SHOULD show, before a person confirms a correction, how it would change the next planned amount of each head that uses the results. Dosing's `maxStep` bounds the change from day to day [OAP-DO Section 12.6].
4. Tests SHOULD end shortly before a dosing day is planned (Section 8.7).
5. Results of a test that used a reagent more than 30 days past its expiry adjust no dosing (Section 9.4.3).

For example, a correction that adds 0.3 dKH to alkalinity results lowers a step method's wanted amount, at a gain of 20 mL per dKH, by 6 mL a day, which a `maxStep` of 5 mL spreads over two days.

13.3 Alarms

1. Accepted results of an analyzer are readings, and manual and laboratory results are results, in the sense of Alarms and Notifications Section 9.3.
2. Suspect, rejected, and unconfirmed results MUST neither raise nor clear range alarms (Sections 10.3.7 and 12.7.2), and the Alarms and Notifications specification counts only accepted results of analyzers [OAP-AN Section 9.3]. A suspect result that is accepted counts from then.
3. A real change in the water can look implausible. With `implausible-result` and `testing-suspended`, a controller SHOULD show the values that were not accepted, and SHOULD suggest a manual test.

13.4 Other Functions

Conditions, interlocks, and modes that read results, such as a kalkwasser reactor that stops while alkalinity is high, or CO2 that stops while the carbonate hardness is low, read them through the same inputs, and so see only accepted results. Results come every few hours at best, so they suit decisions that can wait that long; one maker advises against switching equipment on and off from them.

13.5 Safety Checks

Table 19 — Checks on the use of results

Check	Presence	Applies to	What it requires	Section
Inputs read accepted results	REQUIRED	Controllers	An input that reads an analyzer that a testing function drives reads its latest accepted, corrected result, with its sample time. Rejected and held results never reach an input.	13.1
Range alarms	REQUIRED	Controllers	Only accepted results raise or clear range alarms. A held result that a later result confirms counts from then.	13.3
Results judged once	RECOMMENDED	Controllers, configurations	A dosing head whose input reads results that a testing function screens has no <code>plausible</code> of its own, and a dosing function never sends <code>test</code> to an analyzer that a testing function drives.	13.2

14 Analyzers That Run Their Own Schedules

14.1 Analyzer Programs

1. An analyzer with the `local-program` feature runs the analyzer program in its `program` attribute [OAP-DC Section 10.3], written as JSON text [RFC 8259] without whitespace between tokens. An empty `program` means that the analyzer tests only on command.
2. An analyzer program has the members of Table 20, and MUST be valid against `$defs/analyzerProgram` in `schema/water-testing.schema.json` [JSON Schema].
3. An analyzer MUST reject a write of a program it cannot run in full, and keep its previous program [OAP-DC Section 7.13.2].

Table 20 — Members of an analyzer program

Member	Type	Presence	Description
<code>schedule</code>	array	REQUIRED	The schedule entries (Table 8).
<code>maxDelay</code>	number (s)	OPTIONAL	As in Table 3. Default 3600.
<code>posixTimeZone</code>	string	OPTIONAL	The local time zone, as a POSIX TZ string [POSIX], as in Automation. A controller always writes it (Section 14.3.1); without it, the analyzer uses its own setting.
<code>extensions</code>	object	OPTIONAL	Extensions.

For example, the `program` of a nitrate and phosphate analyzer:

```
{
  "schedule": [
    {
      "parameters": ["nitrate", "phosphate"],
      "times": ["06:30", "18:30"],
      "action": "clean",
      "start": "11:00",
      "days": ["sun"]
    }
  ],
  "posixTimeZone": "AEST-10"
}
```

14.2 Running an Analyzer Program

1. An analyzer with a program MUST follow Sections 8.3 to 8.6 by itself, with its own clock: its testing days and test times, starting tests, delays, and skipped tests.
2. It tests 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], but only while it is enabled.
3. While it is disabled, it starts no test of its own and refuses `test` [OAP-DC Section 10.3]. A disabled analyzer whose `failSafeAction` is `continue` stays disabled when it loses contact, until the `revertAfter` of the command that disabled it falls due, or, for a command without one, until its controller enables it [OAP-FS Section 7.3].
4. It reports each result with a `result` event, and the end of each test, whatever its outcome, with a `testEnded` event [OAP-DC Section 10.3].
5. Without a valid clock [OAP-FS Section 10.5], it starts no scheduled test.
6. With the `reagents` or `waste` feature, it starts no test while a reagent the test uses is empty, or its waste is full.
7. A controller screens, corrects, and records its results, and asks for retests, as for any analyzer.

14.3 Under a Controller

1. For an analyzer whose `runOn` is `analyzer`, the controller writes its analyzer program within a minute of its differing from the analyzer's `program`: its `schedule`, its `maxDelay` if it has one, and `posixTimeZone`. The controller MUST write `posixTimeZone`: the configuration's, or, for a configuration without one, a POSIX TZ string [POSIX] 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, as the Dosing specification has controllers do for head programs [OAP-DO Section 15.3.1].
2. For an analyzer whose `runOn` is `controller`, the controller writes an empty `program`, if the analyzer has the `local-program` feature, so that it does not test twice.
3. The condition `program-not-loaded` holds while an analyzer does not report in its `program` the analyzer program that its controller last wrote, whether it refused the write or has not yet stored it. Once the condition has lasted its delay, 60 s (provisional), which `conditions.json` gives, the controller MUST raise its alarm, and MUST hold the analyzer, so that it does not test on a schedule that is not the configuration's. It writes the program again no more often than every 60 s (provisional), and clears the condition, and ends the hold, once the analyzer reports the program.
4. The controller MUST hold an analyzer with `set enabled false` [OAP-DC Section 10.3], with `revertAfter` for a hold whose end is known [OAP-AU Section 17.5], and without it for a hold whose end is not, such as a suspension (Section 10.5) or the hold of item 3. A held analyzer starts no test, and its tests are delayed as Section 8.6 says.
5. An analyzer whose `failSafeAction` is `continue` stays held when it loses contact, until the hold's `revertAfter` falls due, or, for a hold without one, until its controller enables it (Section 14.2.3).

6. The controller tells whether an analyzer is enabled from its `enabled` [OAP-DC Section 10.3], and, for an analyzer of Device Classes revision 0.2 or earlier, which has no `enabled`, from whether its `state` is `disabled` [OAP-AU Table 5].
7. `delayWhile` and suspension hold the analyzer through its function's demand (Section 7.2), so they act on its program too.
8. For an analyzer without the `reagents` feature, the controller counts its tests from its `testEnded` events, once for each test, whatever its outcome, and, for an analyzer of Device Classes revision 0.1, which has no `testEnded`, from its results: the results of one test share their sample time.

14.4 Size

1. On the module bus, a value holds at most 256 bytes [OAP-MB Section 11.6]. An analyzer program for a module MUST fit in 256 bytes as Section 14.1 encodes it.
2. The Device Protocol carries messages to devices of at least 8192 bytes [OAP-DP Section 9.3], so an analyzer program over the network is not limited this way.
3. The program of Section 14.1 takes 155 bytes, so the limit leaves room for a few more entries, but not for a long list of `times`.

14.5 Safety Checks

Table 21 — Checks on analyzers that run their own schedules

Check	Presence	Applies to	What it requires	Section
The same rules on an analyzer	REQUIRED	Analyzers	An analyzer that runs its own schedule follows Sections 8.3 to 8.6 by itself, with its own clock, whether or not a controller is connected.	14.2
Holding an analyzer	REQUIRED	Controllers, analyzers	A controller holds an analyzer with <code>set enabled false</code> , with <code>revertAfter</code> for a hold that ends, and tells a held analyzer by its <code>enabled</code> . A held analyzer starts no test, and its tests are delayed as on a controller; with <code>failSafeAction continue</code> , it stays held when it loses contact, until the hold's <code>revertAfter</code> falls due.	14.3
Writing analyzer programs	REQUIRED	Controllers	A controller writes each analyzer program to its analyzer, as Section 14.1 encodes it, with <code>posixTimeZone</code> , within a minute of their differing, and an empty program to an analyzer whose schedule the controller runs.	14.3
Programs not loaded	REQUIRED	Controllers	While an analyzer does not report the program its controller last wrote, <code>program-not-loaded</code> holds; once it has lasted its delay of 60 s, the controller raises its alarm, and holds the analyzer until it reports the program.	14.3

15 History

15.1 Results

1. A controller MUST record each accepted result as a Data Model observation [OAP-DM Section 8.4]:
 - `parameter`, `value`, `unit`, and `qualifier`, as the result was reported, corrected;
 - `time`, the sample time, or `sample`, a laboratory result's Sample;
 - `system`, and `vessel`, the function's vessel;
 - `method`: for an analyzer's result, its `method` type and `equipment`, the analyzer's equipment; for a manual result, the method type and `instrument` the person gave;
 - for a corrected result, `correction`: its `raw` result, in the observation's unit, and the `factor` and `offset` applied, the offset in the observation's unit [OAP-DM Section 8.4]; and
 - for a reference result, `reference true` (Section 11.2).
2. Its `extensions["org.openaquariumproject.water-testing"]` holds `$defs/resultRecord` of `schema/water-testing.schema.json`:
 - `function`, and `kind`: `scheduled`, `retest`, `on-demand`, `manual`, or `laboratory`;

- `confirmed: result` for a suspect result that a later result confirmed, or `person` for an entered result a person confirmed; and
 - `expiredReagent: true` for a result of a test that used an expired reagent (Section 9.4).
3. Suspect and rejected results are not observations (Section 10.3.7), and an unconfirmed result is an observation with `unitAssumed true` (Section 12.7).

15.2 Tests

1. A controller SHOULD record each test of an analyzer, and each skipped test, as a Data Model event of the type `analyzer-test` [OAP-DM Section 8.6]:
 - `time`, when the sample was taken, or, for a skipped test, when it fell due; `end`, when its results arrived;
 - `system`, `vessel`, and `equipment`, the analyzer's equipment.
2. Its `extensions["org.openaquariumproject.water-testing"]` holds `$defs/testRecord: function; kind (scheduled, retest, or on-demand); standard, true` for a test of a standard; `action; due; parameters; outcome (completed, failed, skipped, or aborted); results`, each with its `parameter, value, unit, qualifier, raw, status (accepted, suspect, or rejected)`, and, if it was not accepted, the check it failed (`range, jump, first, reagent, or disagree`); the analyzer's `fault`, if any; and `reason`, why it was skipped or failed.
3. A suspect result that a later result settles is recorded in its own test with the status it ends with.
4. Each test is recorded as an event of its own, never as one event that totals several [OAP-DM Section 8.6], because each holds its own results.
5. The results of a test of a standard have no `status`, because they are not screened (Section 8.8.4).

The test record keeps what the observations leave out: every result that was not used, and why.

15.3 Comparisons and Corrections

1. A controller SHOULD record each comparison, each correction applied, and each of the analyzer's own calibrations, as a Data Model event of the type `calibration`, with `equipment` the analyzer's equipment, and a Calibration Record [OAP-DM Section 8.12].
2. A comparison's event holds a Calibration Record [OAP-DM Section 8.12] with the `outcome checked`, its `parameter`, and one point: the `reference` result as its `reference`, the analyzer's raw result as its `reading`, both in the parameter entry's unit, and the reference's sample time as its `time`.
3. A correction's event holds a Calibration Record with the `outcome completed`, its `parameter`, the `model linear`, and, as its `result`, the correction applied, and, as its `previous`, the correction it replaced: the correction's `factor` as `gain`, and its `offset` as `offset`, converted as a difference to the parameter entry's unit (Section 7.5.2) [OAP-DM Section 8.12]. Its `points` MAY list the comparisons it came from, each as in item 2.
4. The event's `extensions["org.openaquariumproject.water-testing"]` holds `$defs/calibrationRecord`, whose `kind` is:
 - `comparison: function`, the raw result's sample time, the reference's `time, method, and sample or observation, difference, and ratio`;
 - `correction: function, method (factor, offset, or person), comparisons`, the number it came from, and `change`, how much it changed the latest result; or
 - `analyzer: the analyzer's own calibration, with function`.
5. An analyzer does not report what its own calibration computed, so the event of one holds a Calibration Record with the `outcome completed or failed`, the `parameter` where the calibration was of one, and no `result` [OAP-DM Section 8.12, item 5].

15.4 Reagents and Waste

1. A controller SHOULD record each replacement of a reagent, and each emptying of the waste, as a Data Model event of the type `maintenance`, with `equipment` the analyzer's equipment.
2. Its `extensions["org.openaquariumproject.water-testing"]` holds `$defs/maintenanceRecord: for kind reagent, the reagent, its expires date and lot, and the share left and estimated when it was replaced; for kind waste, the level estimated when it was emptied`.

15.5 Records of Revision 0.1

Revision 0.1 recorded tests as events of the provisional type `org.openaquariumproject.analyzer-test`, a result's `raw` and `correction` and `reference` in its extension, and comparisons and corrections only in extensions. A consumer reads such records as Data Model Table 3 says, and SHOULD write records of this revision [OAP-DM Section 7.7]. Revision 0.2 recorded the analyzer's own calibration without a Calibration Record, with its `outcome` in the extension; a consumer reads the extension's `outcome` as the outcome of a Calibration Record without a result.

16 Fail-Safe Behavior

16.1 Outputs

Table 22 gives the RECOMMENDED fail-safe configuration [OAP-FS Section 13] of analyzers. It follows the Fail-Safe Behavior recommendation for the category `analyzer` [OAP-FS Table 3].

Table 22 — Recommended fail-safe configuration of analyzers

Analyzer	On loss of contact	After power-up	On battery	Notes
Run by a controller	<code>continue</code>	<code>last</code>	<code>stop</code>	Without the controller, nothing starts its tests.
Running its own schedule	<code>continue</code>	<code>last</code>	<code>stop</code>	It keeps testing, unless its controller held it, until the hold's <code>revertAfter</code> falls due; the controller takes its latest results when it returns.

Testing is harmless to the system, and dosing that uses results stops or returns to its normal amount when the controller is lost [OAP-FS Table 3].

16.2 Lost Inputs

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

Table 23 — What testing does when an input is unavailable

Input	Behavior	Section
A <code>delayWhile</code> condition	Tests wait, then are skipped. <code>input-unavailable</code> is raised.	8.5
<code>waste.full</code>	No test starts. <code>input-unavailable</code> and <code>waste-full</code> are raised.	9.5
The analyzer's device	Tests are delayed, then skipped; results go stale.	7.6, 12.6
The clock	No scheduled test starts; ages are measured as Fail-Safe Behavior Section 10.2.2 says.	8.5

16.3 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. Accepted and suspect results, failures, and suspensions continue from the saved state.
3. Tests that fell due while the controller was off are delayed or skipped as Section 8.6 says, measured from their due times, once the clock is valid.
4. When an analyzer has restarted, which its `uptime` or a `device-restarted` notice shows [OAP-AN Section 13.3], its next result follows a fresh start (Section 10.3).

16.4 A Failed Controller

When the controller fails:

1. analyzers that it runs start no tests;
2. analyzers that run their own schedules keep testing, unscreened, unless the controller held them, and no function uses their results until the controller returns; and
3. when it returns, it takes each analyzer's latest results from `lastResults`, and the `result` events that the device or module kept, and screens them (Section 12.2).

17 Sampling and Placement

These recommendations are for owners and for the makers of analyzers; a maker's own instructions for its analyzer take precedence.

1. The sample intake **SHOULD** be where the water is well mixed, such as near a return pump's intake, below the lowest level the water reaches in normal running, and free of bubbles, as makers advise.
2. It **SHOULD** be away from dosing outlets, and from where top-off water enters, whose water would distort a test (Section 8.7).
3. Makers fix the length of sample lines, and some ask for a recalibration after any change to tubing, because the line's volume and height affect the sample.
4. Waste containers **SHOULD** sit below the analyzer, with their inlet above the liquid so that they cannot siphon back, as one maker requires, and **SHOULD** have a lid, because some reagents give off fumes.
5. Reagents and waste **SHOULD** be kept out of reach of children and pets, and handled as their safety data sheets say.

18 Recommended Values

Table 24 gives the RECOMMENDED values that do not depend on the system. They are provisional, and are published in machine-readable form in `checks.json`. Values that depend on the system and its owner, such as plausibility limits, `maxCorrection`, and reagent `tests`, are set by the owner, from the starting points of Table 25 where there are some.

Table 24 — Recommended values

Area	Value	Recommended	Why
Test schedules	<code>maxDelay</code>	3600 s	A test an hour late still shows the water of that day, and covers a feed mode and a short maintenance mode; a later test falls close to the next one.
Test schedules	Refused test interval	60 s	Asking a refusing analyzer again each minute catches it as soon as it can test, without a stream of refusals.
Consumables	<code>lowDays</code>	14 days	Reagents come from the analyzer's maker, and are usually ordered and shipped rather than bought locally; two weeks leaves time.
Consumables	<code>warnLevel</code>	80 %	Leaves room for about a fifth of the waste container's tests, so emptying it can wait for a convenient time.
Consumables	Expired reagent grace	30 days	An aged reagent biases results slowly, and stopping would leave dosing without results; a month after its expiry, the bias may be more than dosing should follow, while alarms still see the results.
Plausibility	<code>retestDelay</code>	300 s	Lets the analyzer flush its last sample and draw fresh water, while the water itself has barely changed.
Plausibility	<code>maxRetests</code>	2	One retest settles a single bad sample, such as an air bubble; a second settles a retest that is itself bad. More use reagent on a fault a person must fix.
Plausibility	<code>maxFailures</code>	3	Three scheduled tests in a row without a usable result mean a fault, not chance; stopping saves reagent and waste room for when it is fixed.
Plausibility	Short test interval	14400 s	Tests up to 4 hours apart see little real change between them, so a jump as small as the 0.2 dKH at which makers retest points to a bad sample; tests further apart propose the larger <code>maxJump</code> of Table 25.
Corrections	Comparison window	3600 s	A reference sample and an analyzer sample taken within an hour of each other see nearly the same water, if no dose falls between them.
Corrections	Comparisons used	3	The controller proposes a correction only once it has three comparisons within <code>compareDays</code> , from the latest three, so that one wrong reference cannot set it alone.

Area	Value	Recommended	Why
Corrections	<code>compareDays</code>	90 days	Comparing every three months, and after each change of reagent batch, catches the slow drift of reagents and sensors without frequent laboratory costs.
Corrections	Corrections confirmed by size	10 %	Makers advise correcting only differences of 5 % or more; for a parameter without <code>maxCorrection</code> , a correction twice that is worth a look at the analyzer and the reference before a person confirms it.
Results	Results maximum age	2 times	A parameter's default <code>maxAge</code> is twice the longest interval between its scheduled tests, plus <code>maxDelay</code> , so that one failed test does not make its results stale, and two do.
Results	Clock ahead	60 s	An analyzer's clock may drift by up to 60 s and still be valid (Fail-Safe Behavior Section 10.5); a result further ahead of the controller's clock than that has a time that cannot be right.
Analyzers that run their own schedules	Program check	60 s	An analyzer stores and reports a program within seconds; writing it again each minute does not flood it. The delay of <code>program-not-loaded</code> , also 60 s, is in Alarms and Notifications' <code>conditions.json</code> .

Table 25 gives the starting points that a controller proposes for plausibility limits and correction limits (Sections 10.2 and 11.6), published in machine-readable form in `starting-points.json`. Each is for systems of one water type with one system tag [OAP-DM Section 8.1]; a system that matches none has none, and its owner sets its limits. They are provisional, and the owner confirms each before it applies.

Table 25 — Starting points for plausibility limits and correction limits

System	Parameter	Plausible range	<code>maxJump</code> , tests up to 4 h apart	<code>maxJump</code> , further apart	<code>maxCorrection</code>	Basis
marine, reef	alkalinity	4 to 16 dKH	0.2 dKH	0.5 dKH	0.42 dKH	The measuring ranges of analyzers run from about 4.5 to 16 dKH. Analyzers state ± 0.05 to ± 0.1 dKH, and makers retest a result 0.2 dKH from the last; a laboratory states 0.1 dKH.
marine, reef	calcium	250 to 600 mg/L	20 mg/L	40 mg/L	52 mg/L	The measuring ranges of analyzers run from 250 to 600 mg/L or more. Analyzers state ± 6 to ± 15 mg/L; a laboratory states 2 %, 8.4 mg/L of 420 mg/L.
marine, reef	magnesium	850 to 1800 mg/L	50 mg/L	100 mg/L	93 mg/L	The measuring ranges of analyzers run from 850 to 1700 mg/L or more. Analyzers state ± 15 to ± 16 mg/L; a laboratory states 2 %, 27 mg/L of 1350 mg/L.
marine, reef	nitrate	0 to 100 mg/L-NO ₃	2 mg/L-NO ₃	5 mg/L-NO ₃	3.1 mg/L-NO ₃	The measuring ranges of analyzers run from 0 to 40 or 100 mg/L. Analyzers state ± 0.5 to ± 1 mg/L, or 5 % of the result; a laboratory states 2 %, 0.2 mg/L of 10 mg/L.
marine, reef	phosphate	0 to 2 mg/L-PO ₄	0.05 mg/L-PO ₄	0.1 mg/L-PO ₄	0.03 mg/L-PO ₄	The measuring ranges of analyzers run from 0 to 2 mg/L. Analyzers state ± 0.01 mg/L, or 5 % of the result; a laboratory states 2 %, 0.001 mg/L of 0.05 mg/L.

19 Conformance

19.1 Controllers

A conforming controller:

1. runs testing functions as Sections 7 to 13 define, including screening every result before it is used (Section 10.3), applying corrections exactly once (Section 11.5), giving inputs only accepted results (Section 13.1), and using no unconfirmed result and no result whose time is later than its clock (Section 12.7);
2. works with analyzers that run their own schedules as Section 14.3 defines;
3. raises the codes of Table 6 that the checks it implements raise, which include every code of a REQUIRED check, and shows what Section 7.9 requires;
4. records history as Section 15 defines;
5. meets Section 16; and
6. passes the test procedures of Section 19.5 that apply to controllers, for the checks it implements and for every other requirement.

19.2 Analyzers

A conforming analyzer:

1. implements the Device Classes `analyzer` class and the Fail-Safe Behavior specification [OAP-DC Section 13.1];
2. reports each result with the time its sample was taken;
3. meets Section 9.5.6 on returning water to the system;
4. if it has the `correction` feature, applies its corrections as Section 11.5.3 says;
5. if it has the `local-program` feature, runs analyzer programs as Sections 14.1 and 14.2 define; and
6. passes the test procedures of Section 19.5 that apply to analyzers, with no controller connected and again under a controller's control.

19.3 Configuration Documents

A conforming configuration document is a conforming Automation configuration document [OAP-AU Section 20.1] whose testing functions are valid against their schema, and meet every requirement of Sections 7 to 14 that applies to documents, including those a schema cannot express, such as references, parameters, and places.

19.4 Schema and Tools

`schema/water-testing.schema.json` [JSON Schema] is normative for the structure of testing functions, analyzer programs, the saved state, and the water testing members of Data Model observations and events. Where it and this document differ, this document governs.

`checks.json` is normative for the safety checks, the recommended values, and the test procedures, and Tables 9, 12, 14, 16, 18, 19, 21, 24, and 26 are generated from it by `tools/render_spec_tables.py`, which 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 7 is generated from `conditions.json`. Table 25 is generated from `starting-points.json`, whose every `maxCorrection` the same tool checks against its stated accuracies.

The validator, `tools/validate_water_testing.py`, checks configuration documents against the Automation validator, the schema, and the rules a schema cannot express, the analyzer programs of device descriptions, and the water testing records of Data Model documents, of this revision and of revisions 0.1 and 0.2. It warns about choices this Specification advises against, and proposes the starting points of Table 25 for parameters without limits. Its `test_times`, `screen`, `Screen`, `propose_limits`, `propose_correction`, `needs_size_confirmation`, and `default_max_age` functions are reference implementations of Sections 8.3 and 8.4, 10.3, 10.4 and 10.5, 10.2, 11.4, 11.6, and 12.6. It is informative.

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

19.5 Test Procedures

Tests use a controller with simulated analyzers whose results, `state`, `enabled`, reagents, waste, faults, and events the tester can set, whose power the tester can remove, and whose clock the tester can set or stop, and an analyzer 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 9, 12, 14, 16, 18, 19, and 21 has a test in Table 26, 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, which `tools/render_spec_tables.py` checks. A pass criterion that checks what a SHOULD asks applies to a product that does it.

Table 26 — Test procedures

Test	Applies to	Requirements	Method	Pass criteria
Test times (8.4)	Controllers, analyzers	8.3, 8.4, 8.5	Run a schedule that tests alkalinity every 4 hours from 00:00, and calcium and magnesium at 02:00 on Mondays and Thursdays, through a change to and from daylight saving time. Run it on a controller and on an analyzer that runs its own schedule.	Every test falls due within 1 s of the time Section 8.4 gives, on the right days, and the controller and the analyzer agree. Tests that fall due together start one after the other, or together in one <code>test</code> command.
Starting tests (8.5)	Controllers, analyzers	7.6, 8.5, 8.6.2	Make each of these prevent a test from starting when it falls due, one at a time: a clock that is not valid, a maintenance mode, a test already running, a delay condition that is true and then unknown, an empty reagent, a full waste container, and suspended testing. Then make the analyzer refuse <code>test</code> for 5 minutes.	No test starts while any of them holds. Each test starts within 1 s of the last of them ending, if that is within its maximum delay. The refused test raises <code>command-refused</code> , is asked for again no more often than every 60 s, and starts when the analyzer accepts it.
Delays and skipped tests (8.6)	Controllers, analyzers	8.6	With <code>maxDelay</code> 3600 s and tests 4 hours apart, hold the analyzer for 30 minutes from a test’s due time; then for 90 minutes; then for 4.5 hours.	After 30 minutes, the test starts within 1 s of the hold ending. After 90 minutes, it is skipped and recorded, and no test starts when the hold ends. After 4.5 hours, the first test is skipped, and the second starts within 1 s of the hold ending, 30 minutes after it fell due.
Reagent forecasts (9.2)	Controllers	9.1, 9.2	Give a reagent that the analyzer does not track <code>tests</code> of 100, and a schedule of 6 tests a day. Run 40 scheduled tests, 3 retests, and a test of a standard over a week, and set <code>lowDays</code> above the forecast.	The estimate falls by 1 % for each test, retest, and test of a standard, to 56 %, the forecast is about 9 days, and <code>low</code> is raised. Recording a replacement resets the estimate to 100 % and clears <code>low</code> .
Empty reagents (9.3)	Controllers, analyzers	9.3.1, 9.3.2, 9.3.3	Make the analyzer report a reagent empty before a test falls due. Then make it report the reagent empty during a test, and report a result that is plausible.	No test that uses the reagent starts while it is empty, <code>empty</code> is raised, and the tests are delayed, then skipped. The result of the test during which the reagent was empty is rejected, and a retest is asked for; <code>implausible-result</code> is raised only once the scheduled test and its retests have given no accepted result.

Test	Applies to	Requirements	Method	Pass criteria
Consumables nobody tracks (9.3)	Controllers	9.3.6, 9.5.8	Configure an analyzer without the <code>reagents</code> and <code>waste</code> features that tests alkalinity and calcium, with no reagent <code>tests</code> , no <code>waste.tests</code> , and no <code>waste.full</code> . Then give a reagent of alkalinity <code>tests</code> ; then give one of calcium <code>tests</code> , and give the waste <code>full</code> .	The controller warns about alkalinity, calcium, and the waste when the configuration is made. Each warning stays until its parameter has a reagent with <code>tests</code> , or the waste has <code>tests</code> or <code>full</code> , and then clears.
Reagent life (9.4)	Controllers, analyzers	9.4	Set a reagent's <code>shelfLifeDays</code> to 1, record its replacement, and let the clock run past a day. Then record another replacement.	<code>expired</code> is raised after the day, tests continue, their results record the expired reagent, and the second replacement clears it.
Expired reagents and dosing (9.4)	Controllers	9.4.2, 9.4.3, 13.2.5	Give an alkalinity analyzer's reagent an <code>expires</code> date, and a head that plans its dosing from the analyzer's results, with an acceptable range of alkalinity. Let the clock run 29 days past the date, and then 31 days, with results below the range.	At 29 days, the head adjusts its amount from the results. At 31 days, it plans its normal amount, while the results are still recorded with the expired reagent, shown, and raise <code>outside-acceptable-range</code> .
Full waste (9.5)	Controllers, analyzers	9.5.1, 9.5.2, 9.5.3, 9.5.4	Make the waste full by the analyzer's <code>wasteLevel</code> , by the controller's estimate, and by the <code>full</code> condition, in turn. Then make the condition unknown.	No test starts in any case, and <code>waste-full</code> is raised. Testing resumes when the waste is recorded as emptied and the condition is false.
Waste nearly full (9.5)	Controllers	9.5.5	Run tests until the waste estimate passes <code>warnLevel</code> . Then record that the waste was emptied.	<code>waste-nearly-full</code> is raised at the level, testing continues, and the condition clears when the waste is emptied.
No reagent into the system (9.5)	Analyzers	9.5.6	Inspect the analyzer's plumbing and documentation. Then run 10 tests and a cleaning cycle at its default settings, with each outlet led to a container of its own, and with a food dye added to each reagent and to the cleaning solution.	At the default settings, no dye reaches an outlet that returns water to the system. The documentation says where each outlet must go, and, if the owner may return water to the system, what doing so does to the water.
Screening results (10.3)	Controllers	10.3, 16.3.4	With <code>maxRetests</code> 2, give a result outside the plausible range, and a retest near the earlier results; then a result that jumps by twice <code>maxJump</code> , and a retest like it; then a jump, and a retest back near the earlier results; then restart the analyzer and give a result near the earlier ones, and another.	The first is rejected and its retest accepted, so <code>implausible-result</code> is not raised. The first jump is held, and both results are accepted when the retest agrees. The lone jump is rejected. The result after the restart is held until the next one agrees. No rejected or held result reaches an input, a range alarm, or the observations, and the owner sees each with its status.
Retests (10.4)	Controllers	10.4	With <code>maxRetests</code> 2 and <code>retestDelay</code> 300 s, make an analyzer give three implausible results in a row. Then make a test end with a <code>testEnded</code> event whose outcome is <code>failed</code> , with the fault <code>sample-failure</code> , and no result.	Each retest starts 300 s after the result before it, under the same conditions as any test. No more than two retests follow a scheduled test. The failed test is retested in the same way.

Test	Applies to	Requirements	Method	Pass criteria
Repeated failures (10.5)	Controllers	7.7.1, 8.8.5, 9.6.2, 9.6.4, 10.5	With <code>maxFailures</code> 3, make three scheduled tests, and their retests, give only implausible results. Restart the controller. Then clear the latch. Make three scheduled tests fail again, and start a test on demand that gives a plausible result. Then make three scheduled tests in a row be skipped because a reagent is empty, and record its replacement. Then make two scheduled tests give implausible results and a third be skipped because the waste is full, and record that the waste was emptied. Then hold the analyzer with a maintenance mode across three scheduled tests.	After the third, no scheduled test starts, the analyzer is disabled, and <code>testing-suspended</code> is raised; all three remain after the restart. Testing resumes when a person clears it, and when the test on demand gives an accepted result. The three skips caused by the empty reagent suspend testing, and recording the replacement ends the suspension. The suspension with implausible results stays when the waste is emptied, until a person clears it. The three tests skipped during the maintenance mode suspend nothing.
Corrections need a person (11.5)	Controllers	11.5.1, 11.5.2	Enter a correction of the person's own, with a <code>factor</code> of 1.03, as a user with the <code>override</code> role, and then as one with <code>full-control</code> . Then import a laboratory document with a result of a parameter the analyzer measures.	Before the person confirms the correction, the controller shows the previous correction and how much the new one changes the latest result. It applies none until a user with the <code>calibrate</code> operation confirms it, so the user with <code>override</code> cannot. The laboratory result changes no correction.
Corrections applied once (11.5)	Controllers, analyzers	7.5.2, 11.5.3, 11.5.4, 11.5.5, 11.5.7	Apply a correction with a <code>factor</code> of 1.02 and an <code>offset</code> of -0.1 dKH to an analyzer of Device Classes revision 0.3 with the <code>correction</code> feature, to one of revision 0.1 with it, and to one without it. Give each a raw result of 8.0 dKH.	All three results are recorded as 8.06 dKH, with a raw result of 8.0 dKH. Within a minute of the correction being applied, the first analyzer's <code>corrections</code> attribute holds it with the offset -0.1 and the <code>unit</code> dKH, and the second's with the offset -0.0357, in meq/L, the canonical unit.
Comparisons (11.3)	Controllers	11.2.4, 11.2.6, 11.3, 11.4.1, 11.4.4	Record a reference result whose sample was taken 20 minutes after an analyzer's result, and another 2 hours from any of its results. Then record a laboratory result of <code>element:P</code> for an analyzer that tests phosphate. Then record two more laboratory results, a week apart, that each differ from the analyzer's result taken within the comparison window.	The first is compared with that result's raw value, and the comparison is recorded. The second is not compared, and the controller offers to start a test. The result of <code>element:P</code> is not compared with phosphate. No correction is proposed until the third comparison, after which the controller proposes one, with its comparisons.
Correction limit (11.6)	Controllers	11.4.6, 11.6	Set <code>maxCorrection</code> for calcium to 20 mg/L, and record three laboratory results, each about 35 mg/L above the analyzer's. Apply the proposed correction. Then apply a correction of the person's own	<code>correction-limit</code> is raised when the third comparison is recorded, stays while the large correction is applied, and clears when the smaller

Test	Applies to	Requirements	Method	Pass criteria
			within the limit. Then remove <code>maxCorrection</code> , and apply a correction with a factor of 1.12, and then one of 1.08.	one is applied, because applying it ends the proposal. Without <code>maxCorrection</code> , the controller applies the correction of 12 % only once the person has confirmed its size, which it shows, and the correction of 8 % with the person's confirmation alone.
Comparison due (11.7)	Controllers	11.7	Set <code>compareDays</code> to 1, record a comparison, and let the clock run past a day. Then record another comparison.	<code>calibration-due</code> is raised after the day, tests continue, and the second comparison clears it.
Analyzer results (12.2)	Controllers	12.2, 16.4	Make an analyzer report results by events. Disconnect it from the controller during a test, and reconnect it after the result. Then make it report the same result twice. Then make it report <code>clock-invalid</code> and a result, and read its <code>lastResults</code> three times. Then make it report a result 5 minutes ahead of the controller's clock. Then make it report a result of a standard. Repeat the disconnection with an analyzer module on the module bus that runs its own schedule and tests three times while it is disconnected.	Each result is recorded once, with its sample time, including the one read from <code>lastResults</code> after reconnecting, and all three of the module's, which the controller asks it for when contact returns. With the clock invalid, and with the time ahead, the result is recorded once, with the time the controller received it, and the later readings of <code>lastResults</code> add no result. The result of the standard is shown and recorded in its test, and is neither screened nor an observation.
Entered results (12.3)	Controllers	12.3.1, 12.3.2, 12.3.4, 12.3.5, 12.4.1, 12.4.2	As a user allowed <code>enter-results</code> , enter a nitrate result of 10 mg/L-N, a calcium result of 4300 mg/L where 430 was meant, and a calcium result in dKH, and import a laboratory document with a magnesium result outside the plausible range. Then enter a result, and import a laboratory document, as a user not allowed <code>enter-results</code> .	The nitrate is shown as 44.3 mg/L-NO ₃ . The calcium in dKH is refused. The calcium of 4300 mg/L and the magnesium are each recorded only after a person confirms them, and are discarded otherwise. The controller records nothing that the user without <code>enter-results</code> enters or imports.
Laboratory samples imported once (12.4)	Controllers	12.4.5	Import a laboratory document. Import it again. Then import a copy whose Sample has another identifier but the same <code>Laboratory.reference</code> .	The second and third imports add no result, and the controller tells the person that it has the sample already.
Stale results (12.6)	Controllers	12.6.1, 12.6.2, 12.6.3, 12.6.5	With tests every 4 hours, <code>maxDelay</code> 3600 s, and the default <code>maxAge</code> , make every test fail from one point on, with an input of the results whose <code>maxAge</code> is the parameter's. Then let a test succeed.	<code>results-stale</code> is raised 9 hours after the sample time of the last accepted result, and the input becomes unavailable at the same moment. Both clear when a result is accepted.
Reminders to test (12.6)	Controllers	12.6.4	Set <code>remindDays</code> to 7 for nitrate, record a manual result, and let the clock run 7 days. Then enter a result that fails its plausibility check, and discard it. Then enter another result.	<code>test-due</code> is raised after 7 days. The discarded result does not clear it, and the accepted one does.

Test	Applies to	Requirements	Method	Pass criteria
Results with an assumed basis (12.7)	Controllers	12.3.3, 12.4.7, 12.7.1, 12.7.2, 12.7.3, 13.3.2	With an acceptable range of nitrate up to 20 mg/L-NO3 and an <code>observation</code> input of nitrate, enter a nitrate result of 5 from a kit whose basis the person cannot give. Import a laboratory document with a phosphate result whose <code>unitAssumed</code> is <code>true</code> . Then confirm the nitrate as mg/L-N.	Both are shown as waiting for a basis, are recorded with <code>unitAssumed true</code> , and reach no input, range alarm, comparison, or correction, and neither is the latest result. After the confirmation, the nitrate is recorded as 5 mg/L-N without <code>unitAssumed</code> , is shown as 22.1 mg/L-NO3, reaches the input, and raises <code>outside-acceptable-range</code> .
Sample times later than the clock (12.7)	Controllers	12.5.3, 12.7.4	With a valid clock, enter a result whose sample time is a day ahead. Import a laboratory document whose sample was collected a day ahead. Then correct the entered result's time to an hour ago.	Neither is recorded, the controller shows each time and asks for it to be corrected, and the latest result does not change. Once corrected, the entered result is recorded with the corrected time.
Inputs read accepted results (13.1)	Controllers	13.1	Bind an input and a testing function's analyzer to one analyzer, with a correction. Give an accepted result, then a held result, then a rejected result.	The input reads the accepted result, corrected, with its sample time, throughout; its age grows from that time until a later result is accepted.
Range alarms (13.3)	Controllers	13.3.1, 13.3.2	Set an acceptable range for alkalinity. Give a rejected result outside it. Then give a held result outside it, and a retest that confirms it.	The rejected result raises no range alarm. The held result raises <code>outside-acceptable-range</code> only when the retest confirms it.
Results judged once (13.2)	Controllers, configurations	10.4.5, 10.6, 13.2	Configure a dosing head with <code>plausible</code> , whose input reads an analyzer that a testing function drives. Then make the head hold a result back.	The controller warns about the head's <code>plausible</code> . Any retest is sent by the testing function, within its <code>maxRetests</code> .
The same rules on an analyzer (14.2)	Analyzers	14.2.1, 14.2.2, 14.2.4, 14.2.5, 14.2.6	Run the tests for test times, starting tests, and delays on an analyzer with no controller connected, holding it with <code>set</code> and <code>revertAfter</code> from a test tool.	Each passes as it does on a controller.
Holding an analyzer (14.3)	Controllers, analyzers	14.2.3, 14.3.4, 14.3.5, 14.3.6	Start a 30-minute maintenance mode that stops an analyzer of Device Classes revision 0.3, whose <code>failSafeAction</code> is <code>continue</code> and which runs its own schedule, across a test's due time. Stop the controller's evaluations during the mode, and keep them stopped past its end. Repeat with an analyzer of revision 0.2.	The controller holds each with <code>set enabled false</code> and <code>revertAfter</code> , and tells that the first is held from <code>enabled</code> , and the second from <code>state disabled</code> . No test starts while an analyzer is held, though it has lost contact. Each resumes its schedule when <code>revertAfter</code> falls due, and gives the delayed test, which is within its maximum delay.
Writing analyzer programs (14.3)	Controllers	14.3.1, 14.3.2, 14.4.1	In a configuration with a <code>timeZone</code> but no <code>posixTimeZone</code> , change an analyzer's schedule. Then change its program on the analyzer itself. Then move its schedule to the controller.	Within a minute each time, the analyzer's <code>program</code> holds the configuration's program without whitespace, no longer than 256 bytes on the module bus, with a <code>posixTimeZone</code> that gives the same local time as <code>timeZone</code> for every

Test	Applies to	Requirements	Method	Pass criteria
Programs not loaded (14.3)	Controllers	14.3.3	Make an analyzer that runs its own schedule refuse the program its controller writes, for 3 minutes; then accept it. Then make it take 30 s to report a program its controller writes.	date of the next year, and at the end it is empty. 60 s after the first write, <code>program-not-loaded</code> is raised, the analyzer is held with <code>set enabled false</code> without <code>revertAfter</code> , and no test starts. The controller writes the program again no more often than every 60 s. When the analyzer reports the program, the condition clears and the hold ends. The program reported after 30 s raises nothing, and holds nothing.
Invalid configurations	Controllers, configurations	7.2, 7.3, 7.4, 8.2.1, 8.2.2, 8.6.5, 11.5	Import configurations that each break one rule for documents: a testing function with neither analyzers nor parameters; an analyzer whose output is missing from <code>outputs</code> ; an output in <code>outputs</code> with no analyzer; a function with <code>onUnavailable</code> ; a function whose vessel belongs to another system; two entries for one parameter in one place; an entry whose unit is not accepted for its parameter; a schedule entry that tests a parameter the bound analyzer does not measure; 17 schedule entries; a <code>maxDelay</code> of 90 000 s; a correction whose unit is not accepted for its parameter; and a function that fails its schema.	The controller and the validator reject each one, saying why, and the running configuration is unchanged.
Saved state	Controllers	7.2.8, 7.7, 16.3	With a suspect result held, two skipped tests counted as failures in a row, reagents and waste estimated, and a comparison recorded, cut the controller's power, and restore it. Then export the configuration, and import it into another controller. Then import a configuration of revision 0.2, whose testing function has the type <code>org.openaquariumproject.water-testing</code> and its saved state under <code>state.extensions</code> .	The saved state is <code>state.functions</code> of the testing function. After the restart, and on the other controller, the suspect is still held and is settled by the next result, the failures continue from two, still all skipped tests, and the estimates and the time of the comparison are as before. The configuration of revision 0.2 runs as one of this revision, with its saved state.
Starting points	Controllers	10.2.4, 10.2.5, 10.2.6, 11.6.6	Configure a testing function for a system of the water type <code>marine</code> with the tag <code>reef</code> , without <code>plausible</code> or <code>maxCorrection</code> , whose analyzer tests alkalinity every 4 hours, and calcium once a day. Decline the proposals; then confirm them.	The controller proposes the starting points of Table 25: alkalinity 4 to 16 dKH with a <code>maxJump</code> of 0.2 dKH, calcium 250 to 600 mg/L with a <code>maxJump</code> of 40 mg/L, and <code>maxCorrection</code> 0.42 dKH and 52 mg/L. It applies none of them until the owner confirms them, and then all of them.
Visibility	Controllers	7.9	Configure two analyzers with schedules, reagents, waste, and a correction, and record a comparison. Delay one's next test with a delay condition, hold a suspect result of it, suspend the other, and enter a result whose basis the person cannot give.	The controller shows, for each analyzer, every item of Section 7.9: why the test is delayed, the retest the suspect waits for, the forecasts, the comparison and correction, and the

Test	Applies to	Requirements	Method	Pass criteria
				suspension; and, for each parameter, the latest accepted result with its source and age, and the unconfirmed result.
Analyzer programs	Analyzers	14.1	Write an analyzer a program that is not valid against <code>\$defs/analyzerProgram</code> ; then one that is valid but has an entry the analyzer cannot run, such as a <code>calibrate</code> action on an analyzer without the <code>calibrate</code> command; then a program it can run.	The analyzer refuses the first two, and keeps its previous program. It runs the third, as Section 14.2 says.
History	Controllers	15.1, 15.2, 15.3, 15.4	Run tests that give accepted, suspect, rejected, and corrected results, a skipped test, and a test of a standard; enter a manual result and a reference result; record a comparison, a correction, the analyzer's own calibration, a reagent replacement, and an emptied waste container. Export the controller's history, and check it with the validator.	The history is valid, and records each as Section 15 says: accepted results as observations, with <code>correction</code> and <code>reference</code> where they apply; every test as an <code>analyzer-test</code> event of its own, the test of a standard with <code>standard</code> ; the comparison and the correction as Calibration Records, and the analyzer's own calibration as one without a result; and no suspect or rejected result as an observation.

20 Security Considerations

1. **Corrections change every result.** A correction changes what dosing does, so applying one needs the `calibrate` operation, of the `full-control` role [OAP-SA Table 6], and a controller shows its effect first (Section 11.5.2).
2. **Results steer dosing.** Anyone who can enter a manual result, or import a laboratory document, can change what dosing plans. Plausibility checks, the person's confirmation, and dosing's own limits bound what a false result can do, and entering results and importing laboratory documents need the `enter-results` operation, which an owner grants on purpose [OAP-SA Table 6].
3. **Forged laboratory documents.** A document that claims to come from a laboratory may not. A controller SHOULD show where an imported document came from; this revision does not define a way for laboratories to sign their results.
4. **Configuration.** Changing a testing function, its plausibility limits, or `maxCorrection` needs `change-configuration`; starting a test on demand or a test of a standard, `run-action`; recording replaced reagents and emptied waste, `record-maintenance`; entering and importing results, `enter-results`; clearing `testing-suspended`, `clear-latch` [OAP-SA Table 6].
5. **Clearing a suspension from afar.** A person who clears `testing-suspended` remotely restarts an analyzer that nobody has looked at. A controller SHOULD say so, and show the results that were not accepted.
6. **Hidden results.** Screening hides results from the functions that would use them, never from the owner, who sees every result with its status (Sections 7.9 and 10.3.7).
7. **Cloud services.** Many analyzers send their results to their makers' clouds. This Specification needs none, and the Security and Access specification lets nothing leave the owner's network without the owner's consent [OAP-SA Section 16.3].
8. **Privacy.** Results and tests show what an owner keeps and how; 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 an analyzer that runs its own schedule and a controller.	Not started	
An analyzer program written by one implementation has run on an analyzer of another.	Not started	
Every normative reference is publicly available.	Done	
The checks file, starting points file, schema, and examples are published, and the examples pass the validator.	Done	
Every requirement has a test procedure that names it.	Done	Section 19.5.
Experienced aquarists have reviewed the recommended values, and the open questions on plausibility limits and correction limits are decided.	In progress	The Project Lead decided the questions on 17 October 2026 (Sections 10.2 and 11.6, and Table 25); the review by aquarists waits.
The Open Issues appendix is empty, or each open issue is explicitly deferred.	In progress	Revision 0.3 resolves every open issue but issue 15, whose numbers wait on analyzers and owners; issues 16 to 20, 22, 23, and 25 to 27 are deferred beyond revision 1.0.
No unresolved Exclusion Notices for this Specification are recorded in <code>Notices.md</code> .	Done	
The Maintainer has recorded consensus to approve.	Not started	

Appendix B — Examples (Informative)

The `examples` directory contains:

- `reef-analyzer.json`: a reef with an alkalinity, calcium, and magnesium analyzer on the network, scheduled by the controller, with tests that wait after alkalinity doses, reagents and waste that the analyzer tracks, a waste float, a calcium correction from laboratory results, manual nitrate and phosphate tests with weekly reminders, and an alkalinity head that plans its day from the screened results; with the saved state of a suspect result waiting for its retest.
- `nutrient-analyzer.json`: a frag system with a nitrate and phosphate analyzer module that runs its own schedule, whose reagents and waste the controller estimates, and a carbon dosing head that pauses while nitrate is low.
- `manual-testing.json`: a planted freshwater tank with no analyzer, whose owner's tests are checked on entry and reminded, and whose carbonate hardness test stops CO₂ when it is low.
- `analyzer-program.json`: the device description of the nitrate and phosphate analyzer module, of Device Classes revision 0.3, with its analyzer program.
- `history.json`: a Data Model document of revision 0.4 with accepted results, raw and corrected; a result held back and then rejected, a failed test and its retest, a skipped test, and a test of a standard; laboratory results used as references; a manual result whose basis waits for the owner to confirm it; a comparison and a correction as Calibration Records; the analyzer's own calibration, as a Calibration Record without a result; and a reagent replacement and an emptied waste container.
- `minimal.json`: one analyzer, one test a day.

Appendix C — Commercial Features (Informative)

How the water testing features of commercial controllers and analyzers correspond to this Specification.

Table 27 — Commercial features

Feature	In this Specification
Test schedules: tests a day spread evenly, set times, chosen days, intervals	Section 8: <code>every</code> , <code>times</code> , <code>days</code> , and <code>everyDays</code> .
Tests per parameter at different rates	Several schedule entries (Section 8.2).
Tests that run only in some modes, or depend on another output	Automation modes, and <code>delayWhile</code> (Sections 7.6 and 8.5).
An extra test that does not move the schedule	Tests on demand (Section 8.8).
A test of a standard solution	Tests of a standard (Section 8.8.4).
Reagent and waste tracking from entered volumes, with alerts	Section 9: counted by the analyzer or the controller, with forecasts.
No test without enough reagent	Section 9.3.
Automatic retest of a result that differs from the last	Sections 10.3 and 10.4: <code>maxJump</code> , <code>maxRetests</code> , <code>retestDelay</code> .
No action on the first result after a restart	<code>confirmFirst</code> (Section 10.3).
Stopping after repeated bad results or retests	Section 10.5: <code>maxFailures</code> and <code>testing-suspended</code> .
A “no test in 24 hours” error, or a stale-sample notice	<code>results-stale</code> (Section 12.6).
Offsetting or scaling an analyzer to match a laboratory	Section 11: comparisons, and corrections confirmed by a person.
Calibration against a reference solution	The analyzer’s own calibration (Section 11.8).
Dosing driven by analyzer results	Section 13.2, and Dosing Section 12.
Manual test log	Section 12.3.
Laboratory result import	Section 12.4.
Results from other makers’ analyzers	Any analyzer of the <code>analyzer</code> class, through the Device Protocol or the Module Bus.

Appendix D — Open Issues (Informative)

Issues 1 to 27 were open in revision 0.1. The Project Lead decided issue 28 on 15 October 2026. Issues 29 to 33 arose in revision 0.2. On 17 October 2026 the Project Lead froze the scope of revision 1.0, and decided issues 1 to 7, 29 to 31, and 33; revision 0.3 applies the decisions, and resolves issues 8, 11 to 13, 24, and 32, which other specifications settle. Issue 34 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 20, 22, 23, and 25 to 27 are deferred beyond revision 1.0, and are tracked outside this Specification. Issue 15 stays open: its numbers wait on analyzers and owners. 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 plausibility limits.** *Resolved in revision 0.3, as the Project Lead decided on 17 October 2026.*

`plausible` had no defaults, and no published source sets limits beyond which a reef’s or a planted tank’s values must be a wrong measurement. The alternatives were: no defaults, with a controller proposing limits from the owner’s acceptable range [OAP-AN Section 9] widened by a margin; a published table of starting points by water type and system tag, which the owner confirms; or the analyzer’s own measuring range, which analyzers already treat as a limit. Revision 0.3 publishes a table of starting points by water type and system tag, in `starting-points.json` (Table 25), with the candidate values for reefs: from analyzers’ measuring ranges, 4 to 16 dKH, 250 to 600 mg/L, 850 to 1800 mg/L, 0 to 100 mg/L-NO₃, and 0 to 2 mg/L-PO₄, for alkalinity, calcium, magnesium, nitrate, and phosphate in turn. A controller never applies them without the owner’s confirmation, as Alarms and Notifications Section 9.1.4 and Automation Section 18.1.2 treat target values, and proposes `maxJump` from the test interval: 0.2 dKH, 20 mg/L, 50 mg/L, 2 mg/L-NO₃, and 0.05 mg/L-PO₄ for tests up to 4 hours apart, and 0.5 dKH, 40 mg/L, 100 mg/L, 5 mg/L-NO₃, and 0.1 mg/L-PO₄ for tests further apart (Section 10.2).

2. **How much a correction may change a result before a fault is raised.** *Resolved in revision 0.3, as the Project Lead decided on 17 October 2026.* `maxCorrection` had no default (Section 11.6). The alternatives were: a share of the value, such as 5 %, which makers use as the least difference worth correcting; an amount for each parameter, such as one maker's ± 2 dKH limit on its offset; or a multiple, such as three times, of the combined uncertainty of the analyzer and the reference, from makers' stated accuracies. Revision 0.3 publishes an amount for each parameter in Table 25, three times the combined uncertainty of an analyzer and a laboratory, which the owner confirms; for a parameter without one, the person confirms each correction larger than 10 % (provisional) (Section 11.6).
3. **Returning tested water to the system.** *Resolved in revision 0.3, as the Project Lead decided on 17 October 2026: unchanged.* Section 9.5.6 forbids it unless the maker documents that it is harmless and the owner chooses it, and covers every liquid an analyzer adds, cleaning solutions included. Three makers allow a titrator's waste back into the tank, and say it does no harm; three others never return it, and one maker's nutrient reagents are acidic and must be neutralized. The owner's informed choice stays, and is no longer provisional.
4. **Correction by factor or by offset.** *Resolved in revision 0.3, as the Project Lead decided on 17 October 2026: `factor` stays the default.* Most of the makers that say how they match a laboratory correct by a factor; one uses an offset. Near the values a system is kept at, the two give almost the same results, but they differ when the water moves far from them (Section 11.4).
5. **Results from expired reagents.** *Resolved in revision 0.3, as the Project Lead decided on 17 October 2026.* Testing continued with an expired reagent, and its results were used and marked (Section 9.4). Revision 0.3 keeps them recorded, marked, shown, and used by alarms, and, from 30 days (provisional) after the reagent expired, lets them adjust no dosing, which returns to its normal amount (Section 9.4.3).
6. **Corrections only by a person.** *Resolved in revision 0.3, as the Project Lead decided on 17 October 2026: unchanged.* A correction is never applied automatically, even from a laboratory result (Section 11.5). Automatic corrections would track drift without effort, but a wrong reference, or a sample mixed up in the post, would then change dosing.
7. **Suspending testing.** *Resolved in revision 0.3, as the Project Lead decided on 17 October 2026.* Testing stops after three failures in a row, and stayed stopped until a person cleared it (Section 10.5); some analyzers resume after a restart. Revision 0.3 also ends a suspension when a test on demand, which a person starts, gives an accepted result (Sections 8.8.5 and 10.5.5).

For other specifications

8. **Automation.** *Resolved in revision 0.3: the Automation specification's revision 0.3 lists the function type `water-testing`.* The Automation specification was to list `water-testing` among its function types (its Section 11.10), in place of the provisional extension name; give functions a member of the saved state, in place of `state.extensions["org.openaquariumproject.water-testing"]`; allow a function with no outputs, for a testing function that only checks entered results (Section 7.2.3); and say that an `endpoint` input of an analyzer that a testing function drives reads its screened, corrected results (Section 13.1). Its revision 0.3 does so, as the Project Lead decided on 17 October 2026, and this revision uses the function type `water-testing` and `state.functions["<function id>"]` (Sections 7.2 and 7.7), reading the names of revision 0.2 as well (Section 7.2.8).
9. **Device Classes.** *Resolved in revision 0.2.* This Specification asked the `analyzer` class for the raw value of a corrected result, an event for every test, the method type of each parameter, reagents' tests and waste capacity, tests of a standard, a unit for the offset of `corrections`, and `program` defined as the analyzer program of Section 14.1 (Device Classes Appendix D, issue 19). Device Classes revision 0.2 answers each, and revision 0.3 adds `enabled` (its Appendix D, issue 39). This revision uses them: `raw` (Sections 11.5.7 and 15.1), `testEnded` (Sections 10.4 and 14.3.8), `methods` (Table 3), reagents' `tests` and `parameters` and `wasteTests` (Sections 9.1 and 9.5), `test` with `standard` (Section 8.8.4), `corrections`' `unit` (Section 11.5.5), and `enabled` (Section 14.3.6), with the older forms kept only for analyzers of earlier revisions.
10. **Data Model.** *Resolved in revision 0.2.* Revision 0.1 recorded tests as events of the provisional extension type `org.openaquariumproject.analyzer-test`, results' raw values and corrections in extensions, and comparisons, corrections, and reagent replacements as `calibration` and `maintenance` events with extensions (Data Model Appendix C, issue 20). Data Model revision 0.2 answers this with the event type

`analyzer-test`, an observation's `correction` and `reference`, and Calibration Records, and this revision records history with them (Section 15), keeping in its own extension only what is this Specification's alone. Records of revision 0.1 are read as Data Model Table 3 says (Section 15.5).

11. **Alarms and Notifications.** *Resolved in revision 0.1; the suggestion is settled by the Alarms and Notifications specification's revision 0.3.* Revision 0.1 added `results-stale`, `testing-suspended`, `correction-limit`, `waste-nearly-full`, and `test-due to conditions.json` (Table 7), as Alarms and Notifications Appendix D, issue 14, asked. The suggestion that `input-unavailable`, for an input whose analyzer's results are stale or whose testing is suspended, be a consequence of that condition is made by that specification's revision 0.3.
12. **Dosing.** *Resolved in revision 0.1; the suggestion is settled by the Dosing specification's revision 0.3.* Section 10.6 answers Dosing Appendix D, issue 13: results are screened by the rule of Dosing Section 12.4, retests are given by the testing function, and a head that reads screened results leaves out its own `plausible`, as Dosing Section 12.4.5 says. The suggestion that the Dosing specification take a result's status from the testing function that screens it, rather than judge it again, is made by its revision 0.3.
13. **Security and Access.** *Resolved in revision 0.3, as the Project Lead decided on 17 October 2026.* Recording reagent replacements and emptied waste used `run-action`, provisionally, and entering manual results and importing laboratory documents, which can steer dosing, needed no operation. The Security and Access specification's revision 0.3 adds the operations `record-maintenance`, for records that change no output, and `enter-results`, granted on purpose, which this revision uses (Sections 9.6, 12.3, 12.4, and 20).
14. **Module Bus.** *Resolved in revision 0.1.* An analyzer program must fit in a 256-byte register (Section 14.4), and carries its time zone as `posixTimeZone`, as head programs do [OAP-DO Section 15.4]; the Module Bus needs no change.

Details of this Specification

15. **Numbers.** *Open: to be confirmed with analyzers and owners.* `maxDelay`, `retestDelay`, `maxRetests`, `maxFailures`, `lowDays`, `warnLevel`, the comparison window, the comparisons used, `compareDays`, and the default `maxAge` need confirming, as do the 60 s that revision 0.2 adds for a refused test, an analyzer's clock ahead of the controller's, and an analyzer program that is not loaded. Revision 0.3 adds the starting points of Table 25, the 4 hours that divide their `maxJump` values, the 30 days after which an expired reagent's results adjust no dosing, and the 10 % above which a person confirms a correction's size.
16. **Reference solutions.** *Resolved in revision 0.2.* Analyzers are checked against standard solutions of known value, but revision 0.1 had no way to mark a test of one, so its result was taken for the system's water. Device Classes revision 0.2 adds `standard to test`, `result`, and `testEnded`, and this revision never screens, uses, or records as an observation a result of a standard (Sections 8.8.4 and 12.2.3).
17. **Results while the controller was away.** *Resolved in revision 0.3: the Module Bus specification's revision 0.4 keeps them.* Over the Device Protocol, a controller receives the `result` events it missed when it subscribes again, as far back as the device keeps them [OAP-DP Section 11.6]. On the module bus, an analyzer reported only its latest result of each parameter, so results of tests run while the controller was away were lost. Module Bus revision 0.4 has a module keep its recent events with sequence numbers, and a controller read those it missed when contact returns [OAP-MB Section 11.4] (Section 12.2.1).

Decided by the Project Lead

28. **Holding an analyzer.** *Resolved in revision 0.2.* The Project Lead decided on 15 October 2026 that, with `failSafeAction continue`, a held analyzer stays held until the hold's `revertAfter` falls due (Fail-Safe Behavior Appendix C, issue 13), as the test of holding an analyzer expects. Sections 14.2.3 and 14.3.5 say so, and Section 14.3.6 tells a held analyzer by its `enabled`, and by its `state` only for an analyzer of Device Classes revision 0.2 or earlier (Device Classes Appendix D, issue 39).

The following issues arose in revision 0.2:

29. **Consumables that nothing tracks.** *Resolved in revision 0.3, as the Project Lead decided on 17 October 2026: unchanged.* An analyzer without the `reagents` or `waste` feature, whose configuration gives neither reagents' `tests` nor `waste.tests` or `waste.full`, can run out of reagent or fill its waste with nobody

knowing, and one maker's analyzer then reads high. The controller warns the owner (Sections 9.3.6 and 9.5.8), and does not refuse the analyzer's results; the warning is no longer provisional.

30. **Skipped tests as failures.** *Resolved in revision 0.3, as the Project Lead decided on 17 October 2026.* A skipped scheduled test counts as a failure unless a hold, a lost device, or an invalid clock caused it (Section 10.5.2), so that an analyzer that cannot test because a reagent is empty or its waste is full is suspended, as one whose results are rejected is, while a long maintenance mode does not suspend testing. The cost was that an owner who replaced an empty reagent after three skipped tests also had to clear the suspension. Revision 0.3 ends a suspension that only skipped tests caused when a person records a replaced reagent or an emptied waste container (Section 10.5.5), and keeps whether they did in the saved state as `skipsOnly` (Table 5).
31. **The analyzer's own calibration.** *Resolved in revision 0.3, as the Project Lead decided on 17 October 2026.* A Data Model Calibration Record with the outcome `completed` needed its `result`, but an analyzer does not report what its own calibration computed, so such a calibration was recorded only in this Specification's extension. Data Model revision 0.4 lets a completed record leave out `result` for equipment that does not report one, and this revision records the analyzer's own calibration as such a record (Section 15.3.5).
32. **Unconfirmed results in Automation inputs.** *Resolved: the Automation specification's revision 0.3 makes the change; this issue repeats Data Model Appendix C, issue 35.* The Data Model forbids acting on a value whose basis was only assumed (its Section 9.4), and this Specification keeps unconfirmed results from every input (Section 12.7). The Automation specification's `observation` input passes over such observations from its revision 0.3 (Section 13.1.2).
33. **Analyzer programs that do not load.** *Resolved in revision 0.3, as the Project Lead decided on 17 October 2026: unchanged.* An analyzer that refuses its program is held, and so stops testing (Section 14.3.3). A controller that runs the analyzer's schedule itself in its place is deferred beyond revision 1.0, with the other ideas.

The following issue records the choices of revision 0.3:

34. **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 left them open:
 - Table 25 gives starting points for reef systems only, the only systems for which analyzers' measuring ranges and stated accuracies give values; for other systems, a controller proposes limits from the analyzer's measuring range and the owner's ranges, as before (Section 10.2.4);
 - each `maxCorrection` of Table 25 takes a laboratory as the reference, so that a large correction is flagged sooner, while an owner who compares the analyzer with a test kit sees `correction-limit` more often (Section 11.6); and
 - `maxJump` takes Table 25's smaller value for tests up to 4 hours apart and its larger value otherwise, rather than growing with the interval, because a limit that is too tight costs only a retest (Section 10.2.5).