



OPEN AQUARIUM PROJECT SPECIFICATION

Fail-Safe Behavior

PRE-DRAFT

Revision	0.4
Status	Pre-Draft. A candidate document submitted for consideration.
Date	2026-10-18
Identifier	fail-safe-behavior
Authors	Trent Scholl, Project Lead, Open Aquarium Project
Editors	Trent Scholl
License	Community Specification License 1.0

Pre-Draft. This is a proposed initial draft. The Working Group has not approved it, and it will change. Comment by opening an issue with the `fail-safe-behavior` 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 recommendations file published with this Specification is 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 recommendations file under the Apache License 2.0. Every requirement that can be tested has a published test procedure (Section 14.3), so owners and reviewers can verify a product’s claims for themselves.

2.2 Interoperability

Devices and controllers from different makers agree on what happens when contact, power, or an input is lost. Mixing equipment from several makers therefore leaves no gaps in protection. The behavior uses the attributes defined by the Device Classes specification [OAP-DC] and the equipment categories of the Data Model [OAP-DM].

2.3 Local Control

Devices protect their equipment by themselves when their controller, the network, or the internet is unavailable, and devices with their own programs keep running them as their controller last left them. Nothing in this Specification depends on a cloud service.

2.4 Safety

This Specification exists to meet this tenet. It requires a defined outcome for every common failure, safe states chosen for each kind of equipment, deliberate recovery, and controllers whose failure cannot leave devices believing all is well. A stop with no known end, such as one that keeps a return pump from pumping water onto the floor during a leak, holds when its controller fails and through power cuts (Section 8.4). No

safety function depends only on a radio link, or on an output that waits long before it stops by itself (Section 13.4).

2.5 Sustainability

Load shedding during power cuts (Section 9.5) makes battery backups last longer, so smaller batteries serve the same need. Because protection does not depend on any one maker, owners can keep and combine equipment they already have.

Revision History

Revision	Date	Description	Authors
0.1	2026-10-13	Initial Pre-Draft, revised with the findings of the module bus simulator.	Trent Scholl
0.2	2026-10-15	One reading of <code>continue</code> for every class: an endpoint keeps everything last commanded, including <code>enabled</code> , holds, and a pending revert, which still falls due, and runs a local program only while that allows it; entering <code>failSafeState</code> cancels a pending revert; the inactive state of every class as attribute values, and what a <code>failSafeState</code> gives; stops defined, and never delayed; <code>last</code> restores a pending revert's target, lights fade and pumps ramp to a level they restore, a dosing head's hold survives a power cut, and an interrupted operation never resumes; control resumed only by commands that change an output, and kept once resumed during <code>powerOnDelay</code> ; taking over and releasing a device; the outputs of a lost device unknown until they have had time to fail safe; inputs unavailable on <code>hardware-failure</code> ; when a device's clock is valid; ordering fail-safe actions by timeout; lost contact shown on every status light or display; outputs that switch mains start 1 s apart; outputs that keep their state without power; no signal of power-loss mode to devices, and alarm outputs left to their alarm function on battery; safety functions, and the path that uses no radio link that each needs; interlocks in power-loss mode; firmware updates defined by load category; a test procedure for every requirement; references cited by tag; the notes for return and circulation pumps; measurement intervals that sensors report, and analyzers' <code>enabled</code> , with Device Classes revision 0.3; every open issue resolved, decided, or deferred.	Trent Scholl
0.3	2026-10-17	Lights that fade, and pumps that ramp, up to a fail-safe state that lights or runs them after power-up, and to one they enter at any other time from what they show, while they still reach their inactive state at once, as Device Classes revision 0.4 defines; inputs that give a safety function a path that uses no radio link when those connected by wire are enough to stop its output, whatever the others report, so that a second sensor on a radio link takes nothing from a heater's cutoff; a test for each.	Trent Scholl
0.4	2026-10-18	The Project Lead's decisions of 17 October 2026 applied: latched stops, which an endpoint keeps through loss of contact and power cuts, in place of its fail-safe behavior, until a controller ends them, so that a return pump stopped for a leak stays stopped when its controller fails; an output that stops by itself on loss of contact gives a safety function a path that uses no radio link only while its <code>failSafeTimeout</code> is at most 300 s (provisional); a controller's own link to its network counts as radio; bridged devices judged by the Device Protocol's rule until it requires their fail-safe attributes to match their equipment; the equipment category <code>fan</code> , recommended <code>run</code> on loss of contact, <code>fail-safe</code> after power-up, no power-on delay, and <code>stop</code> on battery; outlets whose backup thermostat is their program, as Device Classes revision 0.5 defines, so that an interlock's stop takes it away; and top-off units that carry an additive recommended <code>state</code> with <code>enabled false</code> . Also: entering a fail-safe state replaces a command that a condition holds back; a test for each; references to the final revisions of the other specifications; issue 2 deferred beyond revision 1.0.	Trent Scholl

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	2
2.5	Sustainability	3
	Revision History	3
3	Scope	6
4	References	6
4.1	Normative References	6
4.2	Informative References	6
5	Terms, Definitions, and Conventions	7
5.1	Requirements Language	7
5.2	Terms and Definitions	7
5.3	Notation	7
6	Introduction	8
6.1	Purpose	8
6.2	Design Principles	8
6.3	Failures and Responses	8
7	Contact and Loss of Contact	9
7.1	Contact	9
7.2	Fail-Safe Timeout	9
7.3	Entering Fail-Safe Behavior	9
7.4	Recovery	10
7.5	One Controller at a Time	10
7.6	Loss of a Device	11
8	Inactive and Unpowered States	11
8.1	Inactive State	11
8.2	Stopping Is Always Allowed	12
8.3	Unpowered State	12
8.4	Latched Stops	12
9	Power	13
9.1	Device Power-Up	13
9.2	Staggered Restart	13
9.3	Unstable Power	14
9.4	Controller Start-Up	14
9.5	Power Loss and Battery Operation	14
10	Input Failures	14
10.1	Unavailable Inputs	14
10.2	Stale Readings	14
10.3	Control Functions and Unavailable Inputs	15
10.4	Redundant Inputs	15
10.5	Time	15
11	Device and Controller Faults	15
11.1	Watchdogs and Restarts	15
11.2	Firmware Updates	16
11.3	Endpoint Faults	16
12	Local Control	16
13	Configuration	16
13.1	Setting the Fail-Safe Configuration	16
13.2	Recommended Configurations	16
13.3	Visibility	18
13.4	Safety Functions	18
14	Conformance and Testing	19

14.1 Devices 19

14.2 Controllers 20

14.3 Test Procedures 20

15 Security Considerations 25

Appendix A — Approval Checklist 25

Appendix B — Scenarios (Informative) 25

Appendix C — Open Issues (Informative) 26

3 Scope

This Specification defines what devices and controllers do when:

- a device loses contact with its controller, or a controller loses contact with a device;
- another controller takes over a device;
- power is lost, restored, or unstable;
- an input becomes unavailable;
- a device or controller restarts, stops responding, or updates its firmware; and
- equipment is under manual control on the device itself.

It also defines safety functions and the path that each needs, gives recommended fail-safe configurations for each kind of equipment, and gives test procedures.

This Specification does not define:

- how messages are encoded, which bindings such as the Module Bus [OAP-MB] and Device Protocol [OAP-DP] specifications define;
- how faults are turned into alarms and notifications, which the Alarms and Notifications specification [OAP-AN] defines;
- automation, which the Automation specification [OAP-AU] defines, beyond the requirement that every control function defines its behavior when an input is unavailable; or
- the safety logic of top-off, water changes, and leak response, which the Water Level and Water Change specification [OAP-WL] defines.

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.
- **[OAP-AN]** Open Aquarium Project, "Open Aquarium Alarms and Notifications", Revision 0.3, [specs/alarms-and-notifications/spec.md](#).
- **[OAP-DC]** Open Aquarium Project, "Open Aquarium Device Classes", Revision 0.5, [specs/device-classes/spec.md](#).
- **[OAP-DM]** Open Aquarium Project, "Open Aquarium Data Model", Revision 0.4, [specs/data-model/spec.md](#).
- **[OAP-SA]** Open Aquarium Project, "Open Aquarium Security and Access", Revision 0.3, [specs/security-and-access/spec.md](#).

4.2 Informative References

- **[OAP-AU]** Open Aquarium Project, "Open Aquarium Automation", Revision 0.3, [specs/automation/spec.md](#).
- **[OAP-DO]** Open Aquarium Project, "Open Aquarium Dosing", Revision 0.3, [specs/dosing/spec.md](#).
- **[OAP-DP]** Open Aquarium Project, "Open Aquarium Device Protocol", Revision 0.3, [specs/device-protocol/spec.md](#).
- **[OAP-FP]** Open Aquarium Project, "Open Aquarium Flow Programs", Revision 0.3, [specs/flow-programs/spec.md](#).
- **[OAP-LP]** Open Aquarium Project, "Open Aquarium Lighting Programs", Revision 0.3, [specs/lighting-programs/spec.md](#).
- **[OAP-MB]** Open Aquarium Project, "Open Aquarium Module Bus", Revision 0.4, [specs/module-bus/spec.md](#).
- **[OAP-PM]** Open Aquarium Project, "Open Aquarium Power Module", Revision 0.3, [specs/power-module/spec.md](#).
- **[OAP-WL]** Open Aquarium Project, "Open Aquarium Water Level and Water Change", Revision 0.3, [specs/water-level-and-water-change/spec.md](#).

- **[P0001]** Open Aquarium Project, "Proposal 0001: Reference Controller Architecture", `proposals/0001-reference-controller-architecture.md`.

5 Terms, Definitions, and Conventions

5.1 Requirements Language

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

5.2 Terms and Definitions

Terms defined in the Device Classes specification [OAP-DC], such as device, endpoint, controller, binding, local program, and revert target, have the same meaning here.

Contact: A device has contact with its controller while it keeps receiving valid messages from that controller (Section 7.1).

Supervision message: A message a controller sends only to show that it is still in control.

Fail-safe behavior: What an output endpoint does when it loses contact with its controller, as set by its `failSafeAction` and `failSafeState` attributes.

Inactive state: The state in which an endpoint's equipment is not doing anything, given as values of its state attributes (Table 2). The inactive state is not always the safe state.

Stop: A command that puts an endpoint, or one attribute of its inactive state, in its inactive state, such as `stop`, or `set` with `on false` (Section 8.2).

Hold: A state, set by a command such as `set` with `enabled false`, in which an endpoint's local program, or a unit's own logic, starts nothing [OAP-DC Section 7.13].

Latched stop: A stop with no known end that an endpoint keeps, in place of its fail-safe behavior, until a controller ends it (Section 8.4).

Unpowered state: The state of an output while the device's processor is not controlling it, such as during a restart, during a firmware update, or without power.

Control function: The part of a controller's automation that sets an output from one or more inputs, such as a heater controlled by a temperature sensor.

Interlock: A rule of a controller that holds outputs stopped while a condition is true, such as an interlock of the Automation specification [OAP-AU Section 13].

Unavailable input: An input whose value must not be used for control (Section 10.1).

Lost device: A device from which its controller has received nothing for longer than its binding allows (Section 7.6).

Local control: A manual control on a device itself, such as a button that runs a pump.

Power-loss mode: How a controller runs the system while mains power is lost and the system runs on a battery.

Stop-on-loss equipment: Equipment of a Data Model equipment category [OAP-DM] whose recommended configuration on loss of contact is `stop` in Table 3, such as a heater, a doser, or a CO2 system.

Safety function: Heater and chiller cutoff, top-off shutoff, leak response, or an interlock that stops stop-on-loss equipment (Section 13.4).

Radio link: A connection over radio, such as Wi-Fi, Thread, or Bluetooth, anywhere between a device and its controller, including the controller's own link to its network.

5.3 Notation

Attribute, command, and fault names are those of the Device Classes specification and appear in `monospace`.

References to other Open Aquarium specifications name them, as in “Device Classes Section 7.4”, or use their tags from Section 4, as in “[OAP-DC Section 7.4]”. A section number with a third part refers to a numbered item of that section: “Section 7.3.2” means item 2 of Section 7.3.

6 Introduction

6.1 Purpose

Aquarium equipment fails in ordinary ways: Wi-Fi drops, controllers crash, cables get cut, and the power goes out. In each case some equipment must keep running and some must stop, and which is which depends on the equipment. A return pump should keep running; a heater should stop.

Commercial controllers each handle these failures differently, and owners often find out how only when something goes wrong. This Specification defines, for each common failure, what devices and controllers do, so that a system built from different makers’ equipment fails safely and predictably.

6.2 Design Principles

1. **Every failure has a defined outcome.** No failure leaves equipment in an undefined state.
2. **The safe state depends on the equipment.** Off is not always safe. Each output is configured for what it controls.
3. **Safety lives in the device.** Devices act on their own when their controller is gone.
4. **Stopping is always allowed.** A stop is never refused or delayed.
5. **Recovery is deliberate.** Devices leave fail-safe behavior only when their controller takes control again, after it has checked the current state.
6. **Layers of protection.** Device limits, fail-safe behavior, controller logic, and the equipment’s own protections, such as a heater’s built-in thermostat, each work on their own.
7. **No safety function rests on radio alone.** Every safety function keeps a path that uses no radio link.

6.3 Failures and Responses

Table 1 — Failures and responses

Failure	Detected by	Response	Section
Controller unreachable: network down, controller crashed or stopped responding	Device: no contact for <code>failSafeTimeout</code>	Each output applies its fail-safe behavior, or keeps a latched stop	7.1 to 7.4, 8.4
Another controller takes over	Device: a request of its binding to take control	Every output enters its fail-safe behavior; the new controller resumes control of each	7.5
Device unreachable	Controller: no messages from the device	Its inputs become unavailable, and its outputs unknown until they have had time to fail safe; a fault is raised	7.6
Device or controller restart	Power-up	Power-up behavior; the controller waits for valid configuration, inputs, and time	9.1, 9.4
Mains power lost	Controller power-fail input	Power-loss mode	9.5
Input unavailable: sensor fault, cut wire, stale reading	Device fault or quality; controller maximum age	Control functions apply their unavailable-input behavior	10
Clock not valid	Controller or device	Time-based functions and local programs treat the time as unavailable	10.5
Processor stops running	Watchdog	Restart; outputs are in their unpowered state meanwhile	11.1
Firmware update	—	Outputs keep their state or apply fail-safe behavior	11.2
Manual control on the device	Device	<code>controlSource</code> becomes <code>local</code> ; stopping is still allowed	12
Radio link lost or jammed	Devices and controller, as loss of contact or of a device	Each safety function keeps a path that uses no radio link	13.4

7 Contact and Loss of Contact

7.1 Contact

1. A device tracks contact with its controller for the device as a whole. Each endpoint compares the time since the device last received a valid message from its controller with its own `failSafeTimeout`.
2. A valid message is a command, an attribute write, or a supervision message that the binding identifies as coming from the device's controller, authenticated where Section 15 requires it. A binding MAY leave attribute writes out, for example because a controller that is starting up writes settings before it controls anything.
3. A controller MUST send each device it controls a valid message at least once in every period equal to one third of the shortest `failSafeTimeout` among that device's endpoints.
4. A controller MUST send supervision messages from the same process that runs its control functions, so that a controller whose control functions stop also stops supervising its devices. A communication task MUST NOT send supervision messages on its own.
5. A device MUST measure the time since contact with a clock that does not change when the time of day is set.

Bindings, such as the Module Bus [OAP-MB] and Device Protocol [OAP-DP] specifications, define supervision messages.

7.2 Fail-Safe Timeout

1. `failSafeTimeout` MUST be at least 5 s and at most 3600 s. A device MUST reject a write of a value outside that range, and keep the previous value [OAP-DC Section 7.3].
2. The RECOMMENDED defaults are 30 s for modules on the module bus and 120 s for network devices, which must ride out short Wi-Fi outages. Both are provisional (Appendix C, issue 5).
3. An endpoint MUST begin its fail-safe behavior no earlier than `failSafeTimeout` and no later than `failSafeTimeout` plus 2 s after the device's last contact.

A `failSafeTimeout` longer than 300 s (provisional) leaves an output on a radio link without a path for its safety functions, because a jammed link would leave it running for that long (Section 13.4.3).

7.3 Entering Fail-Safe Behavior

When its timeout expires, an endpoint MUST:

1. if `failSafeAction` is `state`, cancel any pending revert, and any command that a condition holds back [OAP-DC Section 7.8.3], and enter `failSafeState` (Section 8.1);
2. if `failSafeAction` is `continue`, keep everything its controller last commanded, including `enabled`, any hold, and any pending revert, which still falls due [OAP-DC Section 7.4], and run its local program, or a unit its own logic, only while what was last commanded lets it;
3. set `controlSource` to `fail-safe`; and
4. add `communication-lost` to the device's `faults`.

An endpoint that a latched stop holds keeps it in place of steps 1 and 2 (Section 8.4).

Step 4 depends on contact alone. The device reports `communication-lost` while it has had no contact for longer than the `failSafeTimeout` of any of its output or unit endpoints, counting from the end of that endpoint's `powerOnDelay` after power-up, and removes it when contact returns (Section 7.4).

With `continue`, what was last commanded decides whether an endpoint runs its program [OAP-DC Section 7.13]:

- a dosing head, an analyzer, or a top-off unit runs its program or its own logic only while it is enabled, so one that its controller held or disabled stays held or disabled;
- a light, a pump, or an outlet whose backup thermostat is its program [OAP-DC Section 7.13.9] runs its program only if it was following it, and otherwise keeps the levels, the speed, or the state last set; and
- a pending revert that falls due returns the endpoint to its revert target, which may be its program.

Entering `failSafeState` cancels a pending revert, so that a revert target that is not safe never follows the fail-safe state. For example, a heater outlet that its controller held off for a while, with a revert to on, stays off when its controller fails. Entering `failSafeState` likewise replaces a command that a condition holds back, such as a command to switch on a heater outlet that a backup thermostat's cutoff holds off, so that the outlet stays off when the cutoff ends after its controller has failed.

A device that has a status light or a display MUST show that it has lost contact while it reports `communication-lost`. It MAY also sound.

Where outputs must reach their fail-safe behavior in order, such as a calcium reactor's CO2 valve closing before its pump stops, their `failSafeTimeouts` SHOULD differ by at least 3 s, the output that must act first having the shorter, so that the times of Section 7.2.3 cannot overlap. A device MAY apply fail-safe behaviors that fall due together, such as at a takeover or a release (Section 7.5), in an order its manufacturer documents, finishing within 2 s.

`continue` SHOULD be used only for equipment whose last commanded state is always safe to keep, or that runs its own program. Temporary states are not safe states. For example, a return pump configured to `continue` would stay off if its controller failed after a stop with no end that the controller did not latch, such as a power-loss mode's [OAP-AU Section 14.6]. Controllers make temporary states safe by sending them with `revertAfter`, which falls due whether or not the controller is there, and keep a stop that has no known end by latching it (Section 8.4).

7.4 Recovery

1. When contact returns, the device MUST remove `communication-lost` from its `faults`, and each endpoint MUST stay in its fail-safe behavior until its controller resumes control of it.
2. A controller resumes control of an endpoint by sending it a command that changes the endpoint's output [OAP-DC Section 7.4], and that the endpoint accepts, even when the command sets the state the output already has. A command changes the output when it sets, starts, or stops what the endpoint's equipment does, such as `set`, `stop`, `dose`, `prime`, `feed`, `advance`, `fill`, `test`, or `runProgram`. Supervision messages, attribute writes, refused commands, and commands that only record, clear, show, or silence something, such as `identify`, `silence`, `refill`, `resetFault`, and `replaceReagent`, MUST NOT resume control [OAP-DC Section 8.2]: an endpoint carries out such a command, if it accepts it, and stays in its fail-safe behavior. When control resumes, the endpoint MUST set `controlSource` to `controller`.
3. Before resuming control of an endpoint, a controller MUST read the endpoint's current state and evaluate its control functions with current inputs. It MUST NOT send commands that were queued while contact was lost.

7.5 One Controller at a Time

1. A device, and each of its endpoints, MUST follow commands from only one controller at a time: the device's controller.
2. A device MUST accept another controller as its controller only when it has no controller, because its controller released it or an owner removed it [OAP-SA Section 10.6], or when it reports `communication-lost` (Section 7.3). A device with no output or unit endpoints never reports `communication-lost`, and its binding says when another controller can take control of it.
3. When a device gets a new controller, each of its output and unit endpoints that is not yet in its fail-safe behavior MUST enter it at once (Section 7.3, steps 1 to 3). Taking control MUST NOT resume control of any endpoint: the new controller resumes control of each as Section 7.4 requires.
4. When its controller releases a device, each of the device's output and unit endpoints MUST apply its fail-safe behavior at once (Section 7.3, steps 1 to 3), without reporting `communication-lost`.

So every endpoint is in its fail-safe behavior before a new controller commands it, however long its own `failSafeTimeout`, and the new controller checks each before it resumes control (Section 7.4.3). This lets a standby controller take over from a failed one without two controllers ever commanding the same equipment. Bindings and the Security and Access specification [OAP-SA] define how controllers are identified, authorized, take control, and release it.

7.6 Loss of a Device

1. A controller **MUST** detect when it has received nothing from a device for longer than a timeout defined by the device's binding. The device is then lost.
2. It **MUST** then treat all of that device's inputs as unavailable (Section 10), and raise a fault for the device as the Alarms and Notifications specification defines [OAP-AN].
3. It **MUST** treat the state of each of a lost device's outputs as unknown until that output's `failSafeTimeout` plus 2 s after the last valid message it sent the device, because until then the device may still have contact, or be waiting out its timeout (Section 7.2.3). After that, it treats each output as being in its fail-safe behavior. A binding **MAY** define that time otherwise, for example where a device that cannot reach its controller also stops counting the controller's messages as contact.
4. While an output's state is unknown, a controller **MUST NOT** rely on it being in any state: an interlock or a control function that depends on whether the output runs treats that as an unavailable input (Section 10.3).

For example, a controller loses a network power strip 30 s after the strip's last report, and the strip's heater outlet has a `failSafeTimeout` of 120 s. The controller does not treat the heater as stopped, for example to let a chiller start, until 122 s after its last valid message to the strip. Where a binding's supervision messages reach every device at once, the controller keeps sending them to a lost device, so, unless the binding defines otherwise, the device's outputs stay unknown until the controller hears from it again. The module bus defines otherwise: a module that cannot transmit stops counting its controller's messages as contact, and its outputs are unknown until their `failSafeTimeout` plus 5 s after its last Heartbeat [OAP-MB Section 12] (Appendix C, issue 18).

8 Inactive and Unpowered States

8.1 Inactive State

Table 2 gives the inactive state of each class as values of its state attributes, as a `failSafeState` gives them [OAP-DC Section 8.3].

Table 2 — Inactive state of each class

Class	Inactive state	Operations in progress
switch	<code>{"on": false}</code>	—
level-output	<code>{"level": 0}</code>	—
pump	<code>{"running": false}</code>	—
light	<code>{"on": false}</code>	—
heater	<code>{"enabled": false}</code>	—
chiller	<code>{"enabled": false}</code>	—
doser	<code>{"dosing": false}</code> , or, with the local-program feature, <code>{"dosing": false, "enabled": false}</code>	A dose, a prime, or a calibration run stops.
valve	<code>{"open": false}</code>	—
feeder	<code>{"feeding": false}</code>	A feed stops.
auto-top-off	<code>{"enabled": false}</code>	A fill stops.
filter-roller	<code>{"state": "idle"}</code> ; a unit with a fault reports <code>fault</code> instead	An advance stops.
analyzer	<code>{"enabled": false}</code> , with which it reports <code>state disabled</code> ; for an analyzer of Device Classes revision 0.2 or earlier, which has no <code>enabled</code> , <code>{"state": "disabled"}</code>	A test MAY finish.

1. A `failSafeState` **MUST** give a value for each attribute of its class's inactive state.
2. It **MAY** also give values of other state attributes that the class's `set` command sets, such as a pump's `speed` or a light's `levels`, and **MUST NOT** give values of any other attributes.
3. An attribute of the inactive state that the class's `set` command does not set, such as a doser's `dosing` or a feeder's `feeding`, **MUST** have its value of Table 2, because nothing can hold it at any other value.

4. A device MUST reject a write of `failSafeState` that breaks item 1, 2, or 3, or that has a value its `set` command would reject, and MUST keep the previous value [OAP-DC Section 7.3]. For example, a pump rejects `{}` and `{"speed": 70}`, and accepts `{"running": false}` and `{"running": true, "speed": 70}`.
5. Entering `failSafeState` gives each attribute that it lists that value, and stops the operations in progress that Table 2 names.
6. A light or a pump that enters a `failSafeState` that lights or runs it MUST reach each of its levels or speeds evenly, from 0 % after power-up (Section 9.1), and otherwise from its value at that moment, over the times that the Device Classes specification defines [OAP-DC Sections 8.3.5 and 8.3.6]. It MUST reach its inactive state, or any attribute of it that the `failSafeState` gives, at once (Section 8.2). So a pump that its controller paused for feeding ramps up to its fail-safe speed when its controller fails, rather than starting at once at full force.

8.2 Stopping Is Always Allowed

A device MUST accept and carry out, at any time, a valid stop from its controller: a command that puts an endpoint, or one attribute of its inactive state, in its inactive state, such as `stop`, or `set` with `on`, `running`, or `enabled false`. This includes while a local control is in effect, which the stop ends (Section 12), during `powerOnDelay` (Section 9.1), and during a firmware update until its final restart (Section 11.2). Nothing in a device delays or refuses a stop, such as a limit on the requests it handles, or a fade or ramp in progress [OAP-DC Section 7.4].

8.3 Unpowered State

1. The manufacturer MUST document each output's unpowered state.
2. Where practical, an output's unpowered state SHOULD match the fail-safe state it is designed for. For example, an outlet intended for a return pump SHOULD use a relay that is closed when unpowered.
3. A device MUST NOT let an output change state erratically while its processor starts or stops, for example by switching a relay on and off.
4. An output whose switching device keeps whatever state it was last set to without power, such as a latching relay, has no unpowered state, and its manufacturer documents that it keeps its last state (item 1). It SHOULD NOT drive stop-on-loss equipment, which it would keep running if the device's processor stopped while the load still had power.

8.4 Latched Stops

A controller stops some outputs with no known end, such as an interlock that stops a return pump while a leak sensor is wet, or an emergency mode [OAP-AU Sections 13 and 14.7]. With its fail-safe behavior alone, such an output could run again when its controller fails: a return pump whose fail-safe state runs it would pump water onto the floor, with no controller left to see the leak. A controller therefore latches such a stop [OAP-DC Section 7.4.13], and the endpoint keeps it until a controller ends it. The Automation specification says which stops a controller latches [OAP-AU].

1. While a latched stop holds an endpoint, the endpoint MUST keep it, in place of its fail-safe behavior and its `powerOnBehavior`:
 - when it loses contact (Section 7.3), and at a takeover or a release (Section 7.5);
 - after power-up, once its `powerOnDelay` has passed (Section 9.1, steps 2 and 4);
 - during a firmware update (Section 11.2.1); and
 - when a local control ends (Section 12.4).

It still sets `controlSource` to `fail-safe`, and its device still reports `communication-lost` (Section 7.3, steps 3 and 4).

2. An endpoint MUST keep a latched stop across power loss.
3. Only a command from its controller that changes the endpoint's output (Section 7.4.2), other than another latched stop, ends a latched stop. Contact that returns, supervision messages, attribute writes, commands that do not change the output, and a local control do not end it. A local control still takes effect (Section 12), and when it ends, the endpoint returns to the latched stop.

4. Only a stop can be latched, so a latched stop never starts anything: the endpoint keeps stopped what the stop stopped, and nothing more.

While a device's processor does not control an output, such as during a restart, the output is in its unpowered state, latched or not (Section 8.3). So a latched stop does not hold a return pump on an outlet whose unpowered state runs it while the device's electronics are without power.

For example, a leak sensor gets wet, and an interlock stops the return pump with a latched stop. The controller then fails. When the pump's `failSafeTimeout` passes, the pump keeps the latched stop rather than entering its fail-safe state, which would run it, and the alarm output, which runs on loss of contact, calls a person (Table 3). A power cut does not change this: after power-up, the pump stays stopped until a controller ends the latch.

9 Power

9.1 Device Power-Up

After power-up, and after any restart, each output and unit endpoint MUST:

1. set `controlSource` to `fail-safe`, and keep its output in its inactive state for `powerOnDelay`, accepting stops and commands that do not change the output (Section 7.4.2), and refusing others;
2. then apply `powerOnBehavior`: its fail-safe behavior (Section 7.3, steps 1 to 3) if `fail-safe`, or, if `last`, the state it had before it lost power, or, if a revert was pending then, that revert's target;
3. keep `controlSource` `fail-safe` until its controller resumes control (Section 7.4); and
4. if its controller has not resumed control within `failSafeTimeout` after the end of `powerOnDelay`, apply its fail-safe behavior (Section 7.3, steps 1 to 3).

Once its controller resumes control, the endpoint MUST follow its controller, and skip the steps that remain. An endpoint that a latched stop held when it lost power keeps it in place of steps 2 and 4 (Section 8.4). During `powerOnDelay`, only a stop can resume control, and the endpoint MUST keep refusing commands that would take it out of its inactive state until `powerOnDelay` ends. So an endpoint that its controller stops during `powerOnDelay`, for example because of an interlock, is still stopped when `powerOnDelay` ends.

Whatever its `powerOnBehavior`, an operation that the power cut interrupted, such as a dose, a feed, or an advance of a filter roll, MUST NOT resume by itself.

With `last`, an endpoint restores a pending revert's target, not the temporary state [OAP-DC Section 7.4]. A light that restores a level MUST fade up to it, and a pump that restores a speed MUST ramp up to it, rather than stepping, over a time that the Device Classes specification defines [OAP-DC Section 8.3.3].

A light or a pump whose `powerOnBehavior` is `fail-safe` MUST likewise fade or ramp up from 0 % to a `failSafeState` that lights or runs it, beginning when its `powerOnDelay` ends, over the time that the Device Classes specification defines for a level restored without a program [OAP-DC Section 8.3.5]. A stop still ends the fade or the ramp at once (Section 8.2). So a return pump configured to `run` does not start at full force after every power cut.

With `fail-safe`, nothing has been commanded since power-up, and no pending revert survives the power cut. An endpoint whose `failSafeAction` is `continue` runs its local program, if it has one, unless a hold or a latched stop that survives the power cut stops it. A dosing head's hold MUST survive a power cut, as its `dosedToday` does [OAP-DC Section 8.3.4]. Any other endpoint with `continue` MUST stay in its inactive state until its controller resumes control, and so must a top-off unit, whose own logic is not a local program.

9.2 Staggered Restart

After a power cut, all equipment restarts at once. Staggering the restart limits current surges and lets water levels settle.

1. Equipment that can overflow or run dry if it starts before water levels settle, such as a protein skimmer, SHOULD have a `powerOnDelay`. Table 3 gives recommended delays.
2. A device with several outputs that switch mains MUST switch them on at least 1 s apart (provisional) after power-up, even when their `powerOnDelay` is 0, because the inrush currents of several pumps and heaters at

once can trip a breaker or weld a relay. A device with several outputs of lower voltage SHOULD do the same.

9.3 Unstable Power

1. A device MUST detect a supply voltage too low for reliable operation, add `undervoltage` to its `faults`, and either keep its outputs in a defined state or restart cleanly.
2. A device MUST NOT switch an output repeatedly because its supply is unstable.

9.4 Controller Start-Up

After power-up or a restart, a controller MUST NOT command any output until:

1. it has loaded its configuration;
2. each input used by its control functions has a current value, or has been found unavailable (Section 10); and
3. it has determined whether its clock is valid (Section 10.5).

Until then, devices continue their power-up behavior (Section 9.1).

9.5 Power Loss and Battery Operation

1. A controller that can detect the loss of mains power, for example through a power-fail input or a battery backup, MUST enter power-loss mode when mains power is lost.
2. In power-loss mode, the controller MUST put each output in the state the owner has configured for battery operation, and MUST still apply its interlocks, so that, for example, a pump that would run dry stops on battery too [OAP-AU Section 14.6]. Table 3 gives recommended settings: equipment marked `run` keeps running, and other equipment stops. An owner MAY instead leave an output that signals alarms to its alarm function, so that it can signal the loss of mains power [OAP-AN Section 15].
3. In power-loss mode, the controller MAY reduce the speed of pumps that keep running, to make the battery last longer.
4. When mains power returns, the controller MUST leave power-loss mode and restart outputs in the same order and with the same delays as after a power-up (Section 9.2).

Devices are given no other signal of power-loss mode: they follow what their controller commands, such as a battery state, a lower speed, or a cap on the levels of a light's own program [OAP-DC Section 9.4].

10 Input Failures

10.1 Unavailable Inputs

An input is unavailable when:

1. its sensor endpoint reports a `quality` other than `good`;
2. its endpoint reports `wiring-fault`, `sensor-disconnected`, or `hardware-failure`, or, for an `analog-input`, `out-of-range`;
3. its device is lost (Section 7.6);
4. its latest value is older than the maximum age the controller allows for it (Section 10.2); or
5. it is derived from other inputs and one of them is unavailable, unless the derivation is designed to tolerate this (Section 10.4).

10.2 Stale Readings

1. A sensor endpoint MUST set `quality` to `stale` when its latest reading is older than three times its normal measurement interval. Its manufacturer MUST document that interval, and a sensor that implements `measurementInterval` reports in it the interval it uses now, so that a controller can read it [OAP-DC Section 11.1].
2. A controller MUST also check the age of every reading it uses for control, from the reading's `updated` time, against a maximum age. Where `updated` is unknown, for example while the controller's clock is not valid, the

controller uses the time it received the reading. This protects against a device that keeps reporting a frozen value as `good`.

3. The RECOMMENDED default maximum ages are, for continuous sensors, the longest of 60 s, three times the sensor's `measurementInterval`, where it reports one, and three times the longest interval at which the sensor's binding lets it report; and 48 hours for analyzer results.

10.3 Control Functions and Unavailable Inputs

1. Every control function that uses an input MUST define what it does when the input is unavailable.
2. Unless the owner has configured something else, or the specification for that kind of control function defines a more suitable behavior, a control function whose input is unavailable MUST put its output in that output's fail-safe behavior, as if contact had been lost. For example, the Dosing specification [OAP-DO] can return measurement-driven dosing to its normal dose instead of stopping it.
3. While an input is unavailable, the controller MUST raise a fault for it, as the Alarms and Notifications specification defines [OAP-AN].
4. When the input becomes available again, the control function resumes, except where its specification requires a fault to be cleared first.

10.4 Redundant Inputs

1. Where the failure of a single sensor could cause serious harm, controllers SHOULD allow two sensors to serve one control function.
2. For heater control, the RECOMMENDED method is to act on the higher of two temperature readings and raise a fault when they differ by more than a set amount, by default 1 °C.
3. When one of two redundant inputs is unavailable, the control function MAY continue with the other, and MUST raise a fault.

10.5 Time

1. A controller's clock is valid if it has been set from a trusted time source since the controller started, or if it has been kept by a battery-backed clock that has not lost power.
2. While its clock is not valid, a controller MUST treat the time of day and the date as unavailable inputs for its control functions.
3. Controllers SHOULD have a battery-backed clock, so that schedules keep running without a network.
4. A device's clock is valid while it has been set from its controller's time since the device started, or kept by a battery-backed clock that has not lost power, and the largest drift its manufacturer documents, over the time since the clock was last set, cannot have moved it more than 60 s. A device receives its controller's time as its binding defines, authenticated where Section 15 requires it.
5. A device's clock stays valid when the device loses contact with its controller, until its error could exceed that bound, so that it keeps running its local programs (Section 7.3). A binding's rule that a device reports its timestamps as unknown, such as a module's after 3 s without time synchronization [OAP-MB Section 9.4], does not make its clock invalid.
6. While its clock is not valid, a device that needs the time of day, such as one that runs a local program or a schedule, MUST treat the time of day and the date as unavailable for them, and MUST report `clock-invalid` [OAP-DC Section 7.6].

11 Device and Controller Faults

11.1 Watchdogs and Restarts

1. Devices and controllers MUST use a hardware watchdog that restarts the processor if its firmware stops running.
2. After a restart, a device follows Section 9.1, and SHOULD be applying its power-up behavior within 5 s. Its manufacturer MUST document the longest time it takes.

Controllers can detect a device restart from the device's `uptime`.

11.2 Firmware Updates

1. While a device updates its firmware, each output **MUST** either keep its state or apply its fail-safe behavior. During the final restart, outputs are in their unpowered state (Section 8.3).
2. A device **MUST NOT** start a firmware update while one of its endpoints is dosing, filling, or taking part in a water change:
 - dosing is a `doser` running for any reason, including a prime or a calibration run;
 - filling is an `auto-top-off` unit filling, or an output whose `loadCategory` is `ro-di` that is on, open, or running; and
 - taking part in a water change is an output whose `loadCategory` is `water-changer` that is on, open, or running.
3. If an update fails, the device **MUST** return to working firmware.
4. When a controller's own update will interrupt control for longer than its devices' fail-safe timeouts, the controller **SHOULD** tell the owner before starting.

How updates are authorized and verified is defined by the Security and Access specification [OAP-SA].

11.3 Endpoint Faults

When an endpoint detects a fault that stops it from controlling its equipment correctly, such as a hardware failure, it **MUST** report the fault and **SHOULD** put its output in its unpowered state.

12 Local Control

1. A local control takes effect immediately, and the endpoint **MUST** set `controlSource` to `local` while it is in effect.
2. A local control **SHOULD** end automatically after a set time of no more than 2 hours, unless it is designed to be permanent, such as a hardware off switch.
3. While a local control is in effect, a device **MAY** reject commands from its controller to that endpoint, except stops (Section 8.2).
4. When a local control ends, the endpoint **MUST** return to the state its controller last commanded, including a revert that fell due meanwhile, if the controller had control when the local control started and the device has had contact throughout. Otherwise it **MUST** apply its fail-safe behavior (Section 7.3), which keeps a latched stop (Section 8.4), and its controller resumes control with a command (Section 7.4).
5. Local controls do not affect contact (Section 7.1).

13 Configuration

13.1 Setting the Fail-Safe Configuration

1. Each endpoint **MUST** ship with a fail-safe configuration that is safe for the equipment it is designed for, and the manufacturer **MUST** document it. A general-purpose output, such as an outlet, **SHOULD** ship configured to `stop` (Section 13.2).
2. When the owner assigns equipment to an output by setting its `loadCategory`, the controller **SHOULD** propose the recommended configuration for that category (Table 3) and ask the owner to confirm it.
3. A controller **MUST NOT** change an endpoint's fail-safe configuration, power-on behavior, or power-on delay without the owner's confirmation.
4. Controllers **SHOULD** record each change to these settings in the system's history. The Security and Access specification also requires them to record it in their audit log [OAP-SA Section 12.1].

13.2 Recommended Configurations

Table 3 gives the RECOMMENDED configuration for each Data Model equipment category [OAP-DM]. It is also published in machine-readable form in `recommendations.json`.

- **On loss of contact:** `run` means `failSafeAction` state, with `failSafeState` the equipment's normal running state as set by the owner, for example `{"on": true}`, or `{"running": true, "speed": 70}` for a pump at the owner's normal speed. `stop` means `failSafeAction` state, with `failSafeState` the inactive state (Table 2). `continue` means `failSafeAction` `continue`.
- **After power-up:** the value of `powerOnBehavior`.
- **Power-on delay:** the value of `powerOnDelay`.
- **On battery:** the state in power-loss mode (Section 9.5). `run` and `stop` are as on loss of contact. `function` means that the power-loss mode leaves the output to its alarm function, so that it can signal alarms, among them the loss of mains power (Section 9.5.2).

Table 3 — Recommended configuration by equipment category

Equipment category	On loss of contact	After power-up	Power-on delay	On battery	Notes
return-pump	run	fail-safe	—	run	Stopping the return pump stops filtration and gas exchange. Do not use <code>continue</code> : the last command may have been a stop with no end that is not latched, such as a power-loss mode's (Section 8.4). A pump whose <code>loadCategory</code> is <code>return-pump</code> should reject it (Flow Programs Section 16.1).
circulation-pump	run	fail-safe	—	run	Flow keeps the water oxygenated. A pump that runs its own flow program can use <code>continue</code> ; its controller then never drives it with a pattern that stops it (Flow Programs Section 16.2).
air-pump	run	fail-safe	—	run	Aeration keeps the water oxygenated.
heater	stop	fail-safe	—	stop	A heater stuck on can kill livestock within hours, while a tank cools slowly. A heater that regulates itself (the <code>thermostat</code> feature) can use <code>continue</code> , and so can an outlet that its backup thermostat regulates as its program (Device Classes Section 7.13; Power Module Section 11.5).
chiller	stop	fail-safe	—	stop	A chiller that regulates itself (the <code>thermostat</code> feature) can use <code>continue</code> , and so can an outlet that its backup thermostat regulates as its program (Device Classes Section 7.13; Power Module Section 11.5).
light	continue	last	—	stop	A light that runs its own lighting program keeps the day and night cycle.
doser	stop	fail-safe	—	stop	A missed dose is safer than an overdose. A doser that runs its own fixed dosing program can use <code>continue</code> ; its daily limits still apply.
auto-top-off	continue	last	—	stop	For top-off units that run their own logic with their own sensors and limits. A top-off pump on a switched output uses <code>stop</code> . A unit whose water carries an additive, such as kalkwasser, uses <code>failSafeAction</code> state with <code>failSafeState</code> <code>{"enabled": false}</code> , because only its controller enforces the additive's limits (Water Level and Water Change Section 8.16).
protein-skimmer	run	fail-safe	600 s	stop	After a power cut, the sump level rises while the return pump restarts; a delay keeps the skimmer from overflowing.
filter	run	fail-safe	—	stop	A canister filter left off for hours can lose its oxygen and release waste when restarted; check it after a long outage.
reactor	stop	fail-safe	—	stop	Applies to the parts that add chemicals: kalkwasser feed and CO2 supply. A media reactor's pump can use <code>run</code> .
co2-system	stop	fail-safe	—	stop	Without pH control, CO2 can suffocate fish and crash the pH.
ozone-generator	stop	fail-safe	—	stop	Without ORP control, ozone can reach harmful levels.

Equipment category	On loss of contact	After power-up	Power-on delay	On battery	Notes
uv-sterilizer	run	fail-safe	—	stop	
feeder	stop	fail-safe	—	stop	Uncontrolled feeding pollutes the water.
algae-scrubber	continue	last	—	stop	
analyzer	continue	last	—	stop	Testing is harmless. When the controller is lost, dosing driven by test results stops, or returns to its normal amount, as the Dosing specification defines.
ro-di	stop	fail-safe	—	stop	An open supply valve can flood the room.
water-changer	stop	fail-safe	—	stop	An interrupted water change stops. The Water Level and Water Change specification defines how it resumes.
alarm-signal	run	fail-safe	120 s	function	Sounds when its controller fails, but not after every power cut. A contact to another alarm system uses <code>stop</code> instead, with a relay that is open without power, so that it is closed only while all is well (Alarms and Notifications Section 15.2).
fan	run	fail-safe	—	stop	A fan left running cools a tank by a few degrees at most, which its heater can make up, while a fan stopped in a hot room lets the tank overheat. A fan that switches itself on and off by temperature can use <code>continue</code> .
other	stop	fail-safe	—	stop	When in doubt, stop, and check what the equipment needs.

13.3 Visibility

1. Controllers **MUST** show the owner the fail-safe configuration, power-on behavior, and power-on delay of every output.
2. Controllers **SHOULD** warn the owner when an output's configuration differs from the recommendation for its `loadCategory`, when an output has no `loadCategory`, or when stop-on-loss equipment is on an output that keeps its last state without power (Section 8.3.4).

13.4 Safety Functions

A safety function stops equipment that would cause harm if it kept running: a heater or a chiller past its limit, a top-off that has filled enough, equipment that feeds a leak, or stop-on-loss equipment that an interlock holds stopped. A radio link can drop for minutes, be jammed, or deliver messages late, so no safety function depends on one alone. This is design principle 8 of the reference controller [P0001].

1. Every safety function needs at least one path that uses no radio link: a way to stop each output it protects that still works when every radio link fails.
2. A safety function has such a path for an output when protection inside the device that switches the output stops it on the function's condition by itself, with a sensor of its own, such as a heater's or a chiller's own thermostat, a top-off unit's own high-level sensor, or a power module's backup thermostat [OAP-PM Section 11].
3. It also has such a path for an output when both:
 - the output's device is connected to the controller by wire, or the output stops by itself when it loses contact, because its `failSafeAction` is `state`, its `failSafeState` is its inactive state, and its `failSafeTimeout` is at most 300 s (provisional); and
 - as far as the function's inputs go, either the inputs whose devices are connected to the controller by wire are enough for the function to stop the output on its condition, whatever its other inputs report and whether or not they are available, or the function stops the output whenever an input whose device is not connected by wire is unavailable (Section 10.3).
4. A device is connected by wire when it is the controller itself, a module on the module bus [OAP-MB], or a network device connected by Ethernet, as its network settings show [OAP-DP Section 13.3], while the controller, too, is connected to that network by Ethernet. Every other link counts as a radio link, including

the controller's own link to its network over radio (Section 5.2), a link to a bridged device [OAP-DP Section 16], and a link whose kind the controller cannot tell. An output of a bridged device whose `failSafe` is `device` [OAP-DP Section 16.2] counts as one that stops by itself when it loses contact (item 3) once the Device Protocol requires the endpoint's fail-safe attributes to match what its equipment does; until then, the Device Protocol's stricter rule applies, and a bridged device gives a safety function such a path only through protection inside its equipment (item 2) [OAP-DP Section 16.4].

5. A controller MUST check every safety function of its configuration for such a path after it starts, and whenever its configuration, a binding, or an endpoint's fail-safe configuration changes.
6. While a safety function has no such path, the controller MUST show its owners a standing warning that names the function and the outputs without a path, and says what would give them one, such as a wired sensor, or a fail-safe configuration of `stop`. A standing warning stays shown to every owner until its cause is gone, and cannot be dismissed.
7. A controller MUST NOT refuse a configuration because a safety function has no such path.

For example:

- A heater on an outlet reached over Wi-Fi, whose `failSafeState` is off and whose `failSafeTimeout` is 120 s, has a path even if its temperature sensor is on radio too: the outlet stops the heater by itself if its link fails, and the heater function stops it if the sensor's link fails (Section 10.3.2).
- The same outlet with a `failSafeTimeout` of 3600 s has no path: a jammed link would leave the heater on for an hour, and the controller warns until the timeout is 300 s or less.
- The same heater on an outlet configured to `continue`, without a thermostat of its own or a backup thermostat, has no path: if the link fails, it keeps heating, and the controller warns.
- Leak response whose leak sensors are all on radio has no path, because a lost link hides a leak rather than stopping anything, and the controller warns until a wired leak sensor is added [OAP-WL Section 11.1].
- Top-off whose level sensors are on radio has a path if its pump's outlet stops by itself when it loses contact, because the top-off stops while a level sensor is unavailable [OAP-WL Section 8.8].
- A heater on a wired outlet configured to `continue`, without a thermostat of its own, whose function acts on the higher of two temperature sensors, one wired and one on Wi-Fi (Section 10.4.2), has a path: the wired sensor alone stops the heater at its limit, whatever the other reports, and the function continues on the wired sensor if the other is lost (Section 10.4.3). If the function acted on the average of the two instead, it would have no path, because a low reading from the sensor on Wi-Fi, perhaps a late one, could keep the heater on past its limit.
- A heater on an outlet of a power strip connected by Ethernet, configured to `continue`, without a thermostat of its own, whose function's only input is the strip's own temperature sensor, has a path while the controller is connected by Ethernet too, and none while the controller reaches the network over Wi-Fi, because the controller's own link is then a radio link.

A network that carries Ethernet over radio, such as a wireless bridge, looks wired to a controller, so owners should keep devices that carry safety functions off such links. For the same reason, a controller whose network devices carry safety functions should connect to its network by Ethernet, as the Device Protocol advises [OAP-DP].

14 Conformance and Testing

14.1 Devices

A conforming device meets every requirement for devices in Sections 7 to 13: tracking contact and entering fail-safe behavior on time (7.1 to 7.3), deliberate recovery (7.4), one controller at a time (7.5), inactive, fail-safe, and unpowered states, stops, and latched stops (8), power-up and unstable power (9.1 to 9.3), stale readings and its clock (10.2, 10.5), watchdogs, updates, and endpoint faults (11), local control (12), and shipped configuration (13.1). It meets Section 15 as well, and passes the tests of Table 4 for devices.

14.2 Controllers

A conforming controller meets every requirement for controllers in Sections 7 to 13: supervision (7.1), recovery (7.4), one controller at a time (7.5), loss of a device (7.6), start-up (9.4), power-loss mode if it can detect power loss (9.5), input failures and time (10), configuration (13.1), visibility (13.3), and safety functions (13.4). It meets Section 15 as well, and passes the tests of Table 4 for controllers.

14.3 Test Procedures

Each test names the requirements it checks, and whether it is for devices, controllers, or both. Every MUST and MUST NOT of Sections 7 to 15 is checked by at least one test. A pass criterion that checks what a SHOULD asks applies to a product that does it.

Table 4 — Test procedures

Test	For	Requirements	Method	Pass criteria
Supervision interval	Controllers	7.1.3	Record the messages a controller sends a device it controls while it has no commands to send.	A valid message reaches the device at least once in every period of one third of the shortest <code>failSafeTimeout</code> among its endpoints.
Controller stops	Controllers	7.1.4	Stop the controller's control functions while its communication keeps working.	Devices stop receiving supervision messages and apply fail-safe behavior.
Contact clock	Devices	7.1.5, 7.2.3	While a controller supervises a device, move the time of day it gives the device forward 1 hour and back 1 hour, then stop all messages.	No endpoint applies its fail-safe behavior while messages continue; after they stop, each applies it between <code>failSafeTimeout</code> and <code>failSafeTimeout + 2 s</code> .
Timeout range	Devices	7.2.1	Write <code>failSafeTimeout</code> values of 4 s and 3601 s, then 5 s and 3600 s.	The first two are rejected, and the previous value is kept; the last two are accepted.
Loss of contact	Devices	7.1.1, 7.2.3, 7.3	Stop all messages from the controller.	Each output applies its fail-safe behavior between <code>failSafeTimeout</code> and <code>failSafeTimeout + 2 s</code> ; <code>controlSource</code> is <code>fail-safe</code> ; <code>communication-lost</code> is reported from the end of the shortest <code>failSafeTimeout</code> .
Local indication	Devices	7.3	Stop all messages to a device that has a status light or a display, then restore them.	The device shows that it has lost contact from when it reports <code>communication-lost</code> until contact returns.
Revert cancelled by the fail-safe state	Devices	7.3.1	With <code>failSafeAction</code> state and <code>failSafeState {"on": false}</code> , send an outlet that is on a <code>set</code> with <code>on false</code> and <code>revertAfter 600</code> , then stop all messages.	The outlet enters <code>failSafeState</code> , and is still off 600 s after the command.
Held-back commands and the fail-safe state	Devices	7.3.1	With <code>failSafeAction</code> state and <code>failSafeState {"on": false}</code> , while a backup thermostat's cutoff holds an outlet off, send it <code>set</code> with <code>on true</code> , then stop all messages; after the timeout, end the cutoff. Repeat with a top-off unit whose <code>failSafeState</code> is <code>{"enabled": false}</code> , sent <code>set</code> with <code>enabled true</code> while it reports <code>daily-limit</code> .	Each endpoint enters <code>failSafeState</code> , and stays in it when the condition ends: the outlet stays off, and the unit starts no fill.
Pending revert	Devices	7.3.2	Send a <code>set</code> command with <code>revertAfter</code> , then stop all messages, with <code>failSafeAction</code> set to <code>continue</code> .	The endpoint reverts at the end of <code>revertAfter</code> .
Holds with <code>continue</code>	Devices	7.3.2, 7.3.3	With <code>failSafeAction continue</code> , disable a top-off unit, and hold a dosing head's	After the timeout, the unit stays disabled and the head held: no

Test	For	Requirements	Method	Pass criteria
			program, without <code>revertAfter</code> , then stop all messages; then lower the water below the unit's normal sensor, and let a dose of the head's program fall due. Separately, switch off a heater outlet whose backup thermostat was its program with <code>set</code> and <code>on false</code> , then stop all messages, and cool its probe below its <code>setpoint</code> .	fill or dose starts, and <code>controlSource</code> is fail-safe. The outlet stays off.
Supervision does not resume control	Devices	7.4.1, 7.4.2	Restore messages, but send only supervision messages.	Outputs stay in fail-safe behavior; <code>communication-lost</code> is cleared.
Commands that do not resume control	Devices	7.4.2	With an endpoint in its fail-safe behavior and contact restored, send <code>identify</code> , and, where its class has them, <code>silence</code> , <code>refill</code> , and <code>resetFault</code> .	The endpoint carries out each command it accepts, and stays in its fail-safe behavior, with <code>controlSource</code> fail-safe.
Resuming control	Devices	7.4.2	Send each endpoint a command that changes its output, one that sets the state it already has.	Each endpoint follows the command; <code>controlSource</code> is <code>controller</code> .
Recovery checks	Controllers	7.4.3	Disconnect a device just before the controller would command one of its outputs, and reconnect it after the inputs have changed so that the command no longer applies.	The controller reads the endpoint's state before it commands it, and never sends the command that no longer applies.
One controller	Devices	7.5.1	With a device under one controller, send it commands from a second controller that is authorized to control it.	The device refuses them, and its outputs do not change.
Taking over	Devices	7.4.2, 7.5.2, 7.5.3	Give a device two output endpoints, with <code>failSafeAction</code> state and <code>failSafeTimeout</code> 30 s and 120 s. Stop the first controller's messages; 10 s later, take control with a second controller; take control again once <code>communication-lost</code> is reported.	The first attempt is refused. After the second, both endpoints are in their fail-safe behavior at once, with <code>controlSource</code> fail-safe, and each stays so until the second controller sends it a command that changes its output.
Release	Devices	7.5.4	Release a device from its controller.	Each output and unit endpoint applies its fail-safe behavior at once; <code>communication-lost</code> is not reported.
Lost device	Controllers	7.6	Put a heater and a chiller in one exclusive interlock, the heater bound to a network device whose outlet's <code>failSafeTimeout</code> is 120 s. Disconnect that device while the heater runs and the chiller is called for.	Once the binding's timeout has passed, the device's inputs are unavailable and a fault is raised; the controller does not start the chiller until 122 s after its last valid message to the device.
Fail-safe states	Devices	8.1	For an endpoint of each class, write as <code>failSafeState: {}</code> ; a value that leaves out an attribute of Table 2; one that gives an attribute that <code>set</code> does not set a value other than that of Table 2, such as <code>{"dosing": true}</code> ; and a valid state. Then, with the valid state, start an operation, and stop all messages.	The first three are rejected, and the previous value is kept; on loss of contact, the endpoint takes each value of the valid state, and the operations that Table 2 names stop.
Stopping under local control	Devices	7.4.2, 8.2, 12.3	Start a local control, then send a command for the inactive state.	The output stops and <code>controlSource</code> is <code>controller</code> .
Stopping at any time	Devices	8.2	Send a stop during <code>powerOnDelay</code> , during a firmware update before its final restart, and during a fade or a ramp.	The device carries out each stop at once.
Power-up	Devices	9.1	Remove and restore the device's power with no controller present.	<code>controlSource</code> is fail-safe from power-up; outputs stay inactive for <code>powerOnDelay</code> , then

Test	For	Requirements	Method	Pass criteria
				apply <code>powerOnBehavior</code> , then apply fail-safe behavior after <code>failSafeTimeout</code> .
Control during the power-on delay	Devices	9.1	Give an output <code>powerOnDelay</code> 60 s, <code>powerOnBehavior</code> fail-safe, and a <code>failSafeState</code> that runs it. Restore power with a controller present; at 10 s, send a stop; at 30 s, a command that runs the output; at 70 s, the same command again.	The stop is carried out, and <code>controlSource</code> becomes <code>controller</code> ; the output is still stopped after 60 s; the command at 30 s is refused, and the one at 70 s carried out.
Staggered restart	Devices	9.2.2	Give every mains output of a device <code>powerOnDelay</code> 0 and a <code>failSafeState</code> that runs it, then remove and restore its power with no controller present, recording each output.	The outputs switch on at least 1 s apart.
Revert at power-up	Devices	9.1	With <code>powerOnBehavior</code> last, send a temporary state with <code>revertAfter</code> 600, then cut the device's power for 10 s.	After power-up, the output restores the state it had before the temporary state.
Restoring a level	Devices	9.1	With <code>powerOnBehavior</code> last, run a light at 80 %, or a pump at 70 %, then cut its power for 10 s.	After <code>powerOnDelay</code> , the light fades up, or the pump ramps up, to that level over the time that the Device Classes specification defines, rather than stepping.
Running fail-safe states after power-up	Devices	9.1	Give a pump and a light <code>powerOnBehavior</code> fail-safe, <code>powerOnDelay</code> 10 s, and a <code>failSafeState</code> that runs the pump at 70 %, or lights the light at 80 %, then cut their power for 10 s with no controller present. Repeat, and send each a stop during the ramp or the fade.	After <code>powerOnDelay</code> , the pump ramps up, and the light fades up, from 0 % to its fail-safe state over the time that the Device Classes specification defines, rather than stepping. Each stop takes effect at once.
Running fail-safe states on loss of contact	Devices	8.1.6	Give a pump the <code>failSafeState</code> <code>{"running": true, "speed": 70}</code> , and a light one that lights it at 80 %. Stop the pump, set the light to 20 %, and stop all messages. Repeat with the pump at 100 %, and with the <code>failSafeStates</code> <code>{"running": false}</code> and <code>{"on": false}</code> .	After the timeout, the pump ramps up from 0 % to 70 %, and the light fades up from 20 % to 80 %, over the times that the Device Classes specification defines, rather than stepping; from 100 %, the pump ramps down to 70 %. The pump and the light reach their inactive states at once.
Interrupted operations	Devices	9.1	Cut a device's power during a dose, a feed, or an advance, and restore it, once with <code>powerOnBehavior</code> last, and once with fail-safe and <code>failSafeAction</code> continue.	The operation does not resume.
Holds across a power cut	Devices	9.1	With <code>powerOnBehavior</code> fail-safe and <code>failSafeAction</code> continue, hold a dosing head's program, cut its power, and restore it with no controller present.	The head stays held: no dose of its program starts.
Latched stops	Devices	7.3, 8.4, 9.1, 12.4	Give a return pump <code>failSafeAction</code> state, <code>failSafeState</code> <code>{"running": true, "speed": 70}</code> , <code>powerOnBehavior</code> fail-safe, and <code>powerOnDelay</code> 10 s. Stop it with <code>set</code> , <code>running</code> false, and <code>latch</code> true, as for a leak, then stop all messages. After the timeout, run it with its local control, and end the local control. Cut its power for 10 s, restore it with no controller present, and wait for twice its <code>failSafeTimeout</code> . Restore contact, and send supervision	The pump stays stopped after the timeout, with <code>controlSource</code> fail-safe and <code>communicationLost</code> reported; it runs under the local control, and stops when it ends; it stays stopped after power-up and after its fail-safe timeout, and through every message before the <code>set</code> with <code>running</code> true, which runs it. The stop without <code>latch</code> ends the latch: after the second timeout,

Test	For	Requirements	Method	Pass criteria
			messages, an attribute write, and <code>identify</code> ; then <code>set</code> with <code>running true</code> . Repeat, ending with <code>set</code> with <code>running false</code> and without <code>latch</code> , then stopping all messages again.	the pump ramps up to its fail-safe state.
Unstable power	Devices	9.3	Lower the device's supply voltage slowly below its rated minimum, hold it there, raise it, and then vary it around that minimum.	<code>undervoltage</code> is reported while the voltage is too low; each output stays in a defined state, or the device restarts cleanly; no output switches repeatedly.
Controller start-up	Controllers	9.4	Restart a controller with one input of a control function disconnected, and without a valid clock.	It commands no output until it has loaded its configuration, has a current value of every other input, has found the disconnected one unavailable, and has found that its clock is not valid.
Power-loss mode	Controllers	9.5	On a controller that can detect power loss, remove mains power for 10 minutes, with an interlock that holds stopped an output configured to run on battery; then restore it.	The controller enters power-loss mode, and outputs take their battery states, except the one that the interlock holds stopped; when power returns, outputs restart in the order and with the delays of Section 9.2.
Wiring fault	Devices	10.1.2; [OAP-DC Section 11]	Open and then short each supervised input.	<code>wiring-fault</code> is reported each time.
Hardware failure of an input	Controllers	10.1.2, 10.3	Make an input endpoint used by a control function report <code>hardware-failure</code> , and an <code>analog-input</code> used by another report <code>out-of-range</code> .	The controller treats each input as unavailable: each function takes its unavailable-input behavior, and a fault is raised.
Stale reading	Devices	10.2.1	Stop a sensor from producing new readings; repeat after changing its measurement interval, if it can.	<code>quality</code> becomes <code>stale</code> once the latest reading is older than three of the measurement intervals its manufacturer documents, or, after the change, three of the intervals it reports in <code>measurementInterval</code> , as soon as the binding can report it.
Frozen reading	Controllers	10.1.4, 10.2.2	Make a sensor keep reporting one value as <code>good</code> , with an <code>updated</code> that does not change, for longer than the input's maximum age.	The controller treats the input as unavailable once the reading is older than its maximum age.
Unavailable input	Controllers	10.3	Make the input of a control function unavailable.	The output takes its configured unavailable-input behavior and a fault is raised.
Redundant inputs	Controllers	10.4.3	Give a heater function two temperature inputs, and make one unavailable.	A fault is raised, whether or not the function continues with the other input.
Controller clock	Controllers	10.5.1, 10.5.2	Start a controller with no time source and no battery-backed clock, with a schedule that starts an output now.	The output does not start, and time-based functions treat the time as unavailable, until the clock is set.
Device clock	Devices	10.5.4 to 10.5.6	Start a device that runs a local program without giving it the time; then give it the time; then stop all messages from its controller for longer than its <code>failSafeTimeout</code> .	It reports <code>clock-invalid</code> , and runs no program that needs the time, until it has the time; after its controller stops, it keeps running its program, and does not report <code>clock-invalid</code> .
Restart	Both	8.3.3, 11.1	Force a watchdog restart of a device, and of a controller.	The processor restarts. A device's outputs are never in a state other than their current state, their unpowered state, or

Test	For	Requirements	Method	Pass criteria
				their power-up behavior, and its power-up behavior applies within the time its manufacturer documents; a controller follows Section 9.4.
Firmware update	Devices	11.2.1	Update a device's firmware during normal operation.	Outputs keep their state or apply fail-safe behavior until the update completes.
Update refused	Devices	11.2.2	Start a firmware update while a dosing head doses, primes, or runs a calibration, while a top-off unit fills, and while an output whose <code>loadCategory</code> is <code>ro-di</code> or <code>water-changer</code> is on.	Each update is refused.
Failed update	Devices	11.2.3	Update a device with firmware that does not start, and with firmware that never confirms that it works.	Each time, the device returns to its working firmware.
Endpoint fault	Devices	11.3	Simulate a hardware failure of an output endpoint.	The endpoint reports the fault.
Local control	Devices	12	Start and end a local control with contact throughout, then again with contact lost during it.	<code>controlSource</code> is <code>local</code> while it is in effect; afterwards the endpoint returns to the state its controller last commanded the first time, and applies its fail-safe behavior the second time.
Documentation	Devices	8.3.1, 10.2.1, 11.1.2, 13.1.1	Read the device's documentation.	It states each output's unpowered state, or that it keeps its last state, and its shipped fail-safe configuration, each sensor's measurement interval, and the longest time the device takes to apply its power-up behavior after a restart.
Owner's confirmation	Controllers	13.1.3	Set an output's <code>loadCategory</code> , and decline the configuration the controller proposes.	The controller changes no fail-safe configuration, power-on behavior, or power-on delay.
Visibility	Controllers	13.3.1	Look up each output in the controller's apps.	The fail-safe configuration, power-on behavior, and power-on delay of every output are shown.
Safety functions	Controllers	13.4	Bind a heater to an outlet reached over Wi-Fi, with <code>failSafeAction</code> <code>continue</code> , no thermostat of its own, and no backup thermostat; restart the controller; then give the outlet <code>failSafeAction</code> <code>state</code> with its inactive state and a <code>failSafeTimeout</code> of 3600 s; then a <code>failSafeTimeout</code> of 300 s. Separately, configure leak response with leak sensors on Thread only. Separately, bind a second heater to a wired outlet with <code>failSafeAction</code> <code>continue</code> , no thermostat of its own, and no backup thermostat, and have its function act on the higher of two temperature sensors, one on Wi-Fi and one wired; then have it act on their average. Separately, bind a third heater, configured in the same way, to an outlet of a power strip connected by Ethernet, with the strip's temperature sensor as its function's	The configurations are accepted. The controller shows every owner a standing warning for the heater's cutoff, which no owner can dismiss, and which stays after the restart, and with the timeout of 3600 s, until the timeout is 300 s, and one for the leak response. It shows none for the second heater's cutoff while its function acts on the higher reading, and one once it acts on the average. It shows none for the third heater's cutoff while the controller is connected by Ethernet, and one while it uses Wi-Fi.

Test	For	Requirements	Method	Pass criteria
			only input, first with the controller connected by Ethernet, then by Wi-Fi.	
Authentication	Both	15.2	On a network binding, while a device's controller is silent, send the device supervision messages and commands from a party that is not authenticated as its controller.	They do not count as contact and are not carried out, and the device applies its fail-safe behavior on time.
Authorization	Both	15.4	Change a fail-safe configuration as a party without the authorization the Security and Access specification requires for configuration changes.	The change is refused.

15 Security Considerations

1. Anyone who can block communication can force devices into their fail-safe behavior. Fail-safe configurations keep livestock safe for hours, but not indefinitely: a heater held off for a day lets a tank cool, and a doser held off misses its doses. So each lost device raises a fault that reaches the owner [OAP-AN], equipment that regulates itself, such as a heater with its own thermostat or an outlet with a backup thermostat [OAP-PM], can keep regulating with `continue`, and safety functions keep a path that jamming a radio cannot cut (Section 13.4).
2. Network bindings MUST authenticate commands and supervision messages, so that an attacker can neither keep a device out of fail-safe behavior by imitating its failed controller nor take control of a device after it has entered fail-safe behavior (Section 7.5).
3. A wired bus inside the equipment cabinet, such as the module bus, relies on physical access control, because anyone who can reach it can also unplug the equipment. Its binding SHOULD authenticate messages where its physical layer leaves room for it, for example on CAN FD.
4. Changing fail-safe configuration MUST require the same authorization as other configuration changes [OAP-SA].

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 14.3.	Not started	
Every normative reference is publicly available.	Done	
The recommendations file is published and matches Table 3.	Done	
The Open Issues appendix is empty, or each open issue is explicitly deferred.	In progress	Issue 5 waits on the bench tests of the breadboard prototype.
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 — Scenarios (Informative)

How a system configured with the recommendations in Table 3 responds to common failures.

Table 5 — Example scenarios

Scenario	What happens
The home Wi-Fi router restarts and is down for three minutes.	After 30 s, the controller treats the network devices as lost: their inputs are unavailable, and their outputs unknown. After 120 s, network devices apply their fail-safe behavior: wave pumps and lights that follow their own programs keep running them, and a network doser stops. Modules on the wired bus are unaffected. When Wi-Fi returns, the controller reads each device's state and resumes control.
The controller's firmware stops running.	Supervision messages stop. After 30 s, modules apply their fail-safe behavior: the return pump runs, the heater stops, dosers stop, and devices with indicators show the lost contact. The watchdog restarts the controller, which waits for current readings and a valid clock before resuming control.
The controller fails while the return pump is paused for feeding.	The pause was sent with <code>revertAfter</code> , and the pump's fail-safe state is <code>run</code> . The pump restarts after the fail-safe timeout, or at the end of the pause if that comes first, and ramps up to its speed over 30 s.
The controller fails while it holds a heater's outlet off for an hour with <code>revertAfter</code> .	The outlet's fail-safe state is off. Entering it cancels the pending revert, so the heater does not switch on when the hour is up.
A leak sensor gets wet, an interlock stops the return pump, and the controller then fails.	The interlock's stop has no known end, so the controller latched it. The pump keeps it after its fail-safe timeout, and through any power cut, rather than running and pumping water onto the floor, and the alarm output, which runs on loss of contact, calls a person. Only a controller ends the latch.
The controller fails during a water change, while the top-off unit is disabled and a dosing head's program is held, both without <code>revertAfter</code> .	Both use <code>continue</code> . The top-off unit stays disabled, so it does not refill the drained system with fresh water, and the head stays held, until a person or the controller ends the holds.
A standby controller takes over from a failed one.	Once a device reports <code>communication-lost</code> , the standby controller takes control of it, and every output of the device that is not yet in its fail-safe behavior enters it at once. The standby controller reads each endpoint's state, evaluates its control functions, and resumes control of each endpoint with a command.
A two-hour power cut, with a battery backup.	The controller detects the loss of mains power and enters power-loss mode: circulation, air pumps, and the return pump run, perhaps more slowly; the heater, skimmer, and lights stop; interlocks still apply. When power returns, the return pump ramps back to its normal speed without delay, the skimmer starts after 10 minutes, and the lights resume their program.
A short power cut at midday, with no battery backup.	When power returns, every output stays inactive for its <code>powerOnDelay</code> . A light restores its midday level, and fades up to it rather than jumping. The return pump, whose fail-safe state runs it, ramps up to its speed over 30 s. A dosing head whose program was held stays held, and a dose that the cut interrupted does not resume.
A float switch cable is cut while the top-off pump is running.	The supervised input reports <code>wiring-fault</code> . The level input is unavailable, so the top-off function puts the pump in its fail-safe state, <code>stop</code> , and raises a fault.
The heater's temperature probe is unplugged.	The sensor reports <code>sensor-disconnected</code> . With one probe, the heater function stops the heater and raises a fault. With two probes, it continues on the remaining probe and raises a fault.

Appendix C — Open Issues (Informative)

Issues 1 to 12 were open in revision 0.1. The Project Lead decided issues 8, 11, and 13 to 16 on 15 October 2026, and every other issue later that day, accepting the recommendation in each. Issue 22 arose from revision 0.2 of the Flow Programs specification, and the Project Lead decided it on 16 October 2026, accepting the recommendation; issue 23 arose from the breadboard firmware, and the Project Lead decided it on 17 October 2026; revision 0.3 applies both. Issue 24 arose in revision 0.3, and the Project Lead decided it on 17 October 2026, accepting the recommendation. Issues 25 to 28 record the Project Lead's decisions of 17 October 2026 that revision 0.4 applies. Each issue now says how a revision resolves it, which specification is to apply it, or what it waits for; only issue 5 is open. Issue numbers do not change, because other specifications refer to them.

1. **Standalone backup control.** *Resolved in revision 0.2.* A device runs such a rule as a local program, defined by the specification of its kind of equipment, with `failSafeAction continue`, as an outlet runs the backup thermostat of the **Power Module** specification (its Section 11). Such a rule is protection inside the device for

a safety function (Section 13.4.2). No general rule format is defined, because each rule needs safety requirements of its own, which belong with its equipment. As the Power Module specification asked (its Appendix E, issue 9), Section 8.3.4 says that a latching relay has no unpowered state, and Section 9.2.2 requires outputs that switch mains to start 1 s apart.

Issue 2 is deferred beyond revision 1.0, and is tracked outside this Specification.

3. **Devices on battery.** *Resolved in revision 0.2: no signal is added.* A controller already tells each device what to do in power-loss mode: a battery state, a lower speed, or a cap on the levels of a light's own program (Section 9.5). A separate signal would let devices act on their own idea of battery operation, which the owner could neither see nor configure. A device that runs on its own battery reports `powerSource` [OAP-DC Section 8.1].
4. **Ordering within a device.** *Resolved in revision 0.2.* Outputs that must act in order get `failSafeTimeouts` at least 3 s apart, so that the times of Section 7.2.3 cannot overlap, and a device may order fail-safe behaviors that fall due together (Section 7.3). No new attribute is needed.
5. **Default timeouts.** *Open: waits on the bench tests of the breadboard prototype.* The recommended timeouts of 30 s and 120 s (Section 7.2.2), and the 1 s between outputs that switch mains (Section 9.2.2), stay provisional until they are measured: how long ordinary Wi-Fi outages last, how long a module takes to restart, and the inrush currents of common pumps and heaters. So does the longest `failSafeTimeout` with which an output gives a safety function a path that uses no radio link, 300 s (Section 13.4.3; issue 25), which the timeouts that owners need will confirm.
6. **Local indication.** *Resolved in revision 0.2.* A device that has a status light or a display must show that it has lost contact (Section 7.3). A device without one need not add one, and sounding stays optional, because short network outages are common.
7. **Certification.** *Resolved: yes.* The project's governance says that a compatibility program, if created, will require conformance testing (its Section 5), and the tests of Table 4 are those of this Specification. Which products may carry a mark, and how their results are checked, belong to the program, not to this Specification.
8. **Pending reverts at power-up.** *Resolved in revision 0.2.* With `powerOnBehavior last`, an endpoint that loses power while a `revertAfter` is pending restores the revert's target, not the temporary state (Section 9.1, step 2), as Device Classes revision 0.2 says (its Section 7.4). Raised by the `Automation` specification (Appendix D, issue 6).
9. **Interlocks on battery.** *Resolved in revision 0.2.* A controller applies its interlocks in power-loss mode, so that, for example, a pump that would run dry stops on battery too (Section 9.5.2). Raised by the `Automation` specification (Appendix D, issue 8).
10. **Alarm outputs.** *Resolved in revision 0.2, with Data Model revision 0.3.* The Project Lead decided on 15 October 2026 that the Data Model add the equipment category `alarm-signal`: equipment that signals alarms to people or to other systems, such as a siren, a strobe, an indicator light, or a contact to a house alarm panel. Data Model revision 0.3 adds it, and Table 3 recommends for it `run` on loss of contact, `fail-safe` after power-up, a power-on delay of 120 s, and, on battery, a new value, `function`: the power-loss mode leaves the output to its alarm function (Sections 9.5.2 and 13.2). Its note says that it sounds when its controller fails, but not after every power cut, and that a contact to another alarm system uses `stop` instead, with a relay that is open without power, so that it is closed only while all is well. The Matter Mapping specification gives the category the policy `none`, so that home apps neither sound nor silence it (its Table 10), and the recommendations of Alarms and Notifications Section 15.2 are no longer provisional. Raised by the `Alarms and Notifications` specification (Section 15.2; Appendix D, issue 4).
11. **Lights the controller drives.** *Resolved in revision 0.2: Table 3 is unchanged.* Table 3 recommends `continue` for every light. A light that runs its own program then keeps its day and night, but a light that its controller drives level by level keeps its last level: midday light through the night, or darkness through the day. The Project Lead decided to keep the recommendation, as the `Lighting Programs` specification does, whose controllers send scenes and other temporary states with `revertAfter` (its Section 17.2). A light that restores a level after power-up fades up to it (Section 9.1; issue 15). Raised by the `Lighting Programs` specification [OAP-LP] (Appendix D, issue 7).

12. **Pumps the controller drives, and return pumps.** *Resolved in revision 0.2.* The notes of Table 3 now say what the [Flow Programs](#) specification [OAP-FP] requires: a controller never drives a pump on [continue](#) with a pattern that stops it, and a pump whose [loadCategory](#) is [return-pump](#) should reject [continue](#) (its Sections 16.1 and 16.2). The recommended defaults are unchanged. Raised by the Flow Programs specification (Appendix D, issue 7).

Decided by the Project Lead

13. **[continue](#), holds, and pending reverts.** *Resolved in revision 0.2.* For every class, entering [failSafeState](#) cancels any pending revert (Section 7.3, step 1). With [failSafeAction](#) [continue](#), an endpoint keeps everything last commanded, including [enabled](#), holds, and a pending revert, which still falls due, and runs a local program only while what was last commanded lets it (Section 7.3, step 2). With [powerOnBehavior](#) [last](#), an endpoint restores a pending revert's target (Section 9.1; issue 8). Table 2 gives the inactive state of every class as attribute values, a [doser](#)'s included: [dosing](#) [false](#) and, with [local-program](#), [enabled](#) [false](#); and Section 8.1 says what a [failSafeState](#) gives, and which values a device rejects. The Device Classes, Dosing, Water Level and Water Change, and Water Testing specifications follow it in their revisions 0.3, 0.2, 0.2, and 0.2.
14. **Safety functions.** *Resolved in revision 0.2.* Section 5.2 defines a safety function: heater and chiller cutoff, top-off shutoff, leak response, or an interlock that stops stop-on-loss equipment. Section 13.4 says that each needs a path that uses no radio link: wired to its controller, or protection inside the device, such as its own thermostat, its own high-level sensor, a backup thermostat, or failing safe on loss of contact. A controller shows its owners a standing warning while a safety function has no such path, and does not refuse the configuration. The Security and Access specification (its Section 10.9.4) and the Automation specification (its Section 13.7) refer to Section 13.4.
15. **Restoring a level after power-up.** *Resolved in revision 0.2.* An endpoint that restores a level after power-up with [powerOnBehavior](#) [last](#) fades a light up to it, and ramps a pump up to it, rather than stepping (Section 9.1). Device Classes revision 0.3 defines how long the fade and the ramp last (its Section 8.3.3).
16. **The note for analyzers.** *Resolved in revision 0.2.* The note for [analyzer](#) in [recommendations.json](#), and so in Table 3, says that when the controller is lost, dosing driven by test results stops, or returns to its normal amount, as the Dosing specification defines. The recommended defaults are unchanged.

The following issues arose in revision 0.2:

17. **Measurement intervals.** *Resolved in revision 0.2, with Device Classes revision 0.3.* Section 10.2.1 relies on each sensor's measurement interval, which its manufacturer documents. Device Classes revision 0.3 adds [measurementInterval](#) to [sensor](#): the interval, in s, at which the sensor normally measures, an OPTIONAL state attribute, because some sensors change it. Section 10.2 names it, so that a controller can read it, and set maximum ages from it (Section 10.2.3).
18. **Lost devices on a shared supervision message.** *Resolved in Module Bus revision 0.2 (its Sections 12.3 and 12.4).* On the module bus, the controller's Heartbeat reaches every module, so by Section 7.6.3 the outputs of a lost module stay unknown until the controller hears from it again. The Project Lead decided on 15 October 2026 that a module that has not transmitted a Heartbeat successfully for 3 s, for example because it is in the bus-off state, stops counting the controller's messages as contact, and Module Bus revision 0.2 says so, with a test. A controller then treats the outputs of a lost module as unknown until each output's [failSafeTimeout](#) plus 5 s after the last Heartbeat it received from the module: the 3 s, and the 2 s of Section 7.2.3, as Section 7.6.3 lets a binding define.
19. **The return pump's note.** *Resolved in revision 0.2.* The note for [return-pump](#) in Table 3 gave a feeding pause as the reason not to use [continue](#). A pause is sent with [revertAfter](#), which now falls due with [continue](#) too (Section 7.3), so the reason that remains is a stop with no end, such as an emergency mode's (Automation Appendix D, issue 17), and the note now says so. The default is unchanged.
20. **Analyzers' [enabled](#).** *Resolved in revision 0.2, with Device Classes revision 0.3.* Section 7.3 lets an analyzer with [continue](#) run its program only while it is enabled, and a controller resumes control of an analyzer by sending [set](#) with its current [enabled](#) (Device Classes Section 7.13), but [analyzer](#) reported only [state](#), not [enabled](#). Device Classes revision 0.3 adds an [enabled](#) state attribute to [analyzer](#), and Table 2 gives an analyzer's inactive state as `{"enabled": false}`, with which it reports [state disabled](#).

21. **Numbers to confirm.** *Resolved in revision 0.2.* The Project Lead confirmed the 60 s bound on the error of a device's clock (Section 10.5.4). A schedule a minute out harms nothing in an aquarium, and a clock that drifts 50 parts per million, as many do, reaches the bound only after about two weeks without its controller. The timeouts, and the 1 s between outputs that switch mains, wait for the bench tests (issue 5).

The following issue arose from another specification after revision 0.2:

22. **Running fail-safe states after power-up.** *Resolved in revision 0.3, with Device Classes revision 0.4 (its Appendix D, issue 46), as the Project Lead decided on 16 October 2026.* Section 9.1 made a light that restores a level fade up to it, and a pump that restores a speed ramp up to it, but an endpoint whose `powerOnBehavior` is `fail-safe` entered its `failSafeState` at once, so a pump with the recommendation of Table 3, `run`, stepped from a stop to its fail-safe speed after every power cut. Section 9.1 now says that a light or a pump whose `powerOnBehavior` is `fail-safe` fades or ramps up from 0 % to a `failSafeState` that lights or runs it, beginning when its `powerOnDelay` ends, over the times that Device Classes Section 8.3.5 defines, and that a stop still ends the fade or the ramp at once; the test "Running fail-safe states after power-up" checks it. Issue 24 extends this to such a state entered at any other time, such as on loss of contact. Raised by the [Flow Programs](#) specification (Appendix D, issue 18).

The following issue arose from the breadboard firmware after revision 0.2:

23. **Redundant inputs of a safety function.** *Resolved in revision 0.3, as the Project Lead decided on 17 October 2026.* Section 13.4.3, read strictly, gave a heater with one temperature sensor on a radio link and one wired no path that uses no radio link, because the function does not stop the heater when the radio sensor is unavailable; yet the heater acts on the higher of its two readings (Section 10.4.2), so the wired sensor alone cuts it off. Section 13.4.3 now says that a safety function has such a path for an output, as far as its inputs go, when the inputs of devices connected by wire are enough for it to stop the output on its condition, whatever its other inputs report and whether or not they are available; or when it stops the output whenever an input whose device is not connected by wire is unavailable. Its examples now include such a heater, and the test "Safety functions" checks one. The breadboard controller already checks its safety functions this way.

The following issue arose in revision 0.3:

24. **Running fail-safe states on loss of contact.** *Resolved in revision 0.3, with Device Classes revision 0.4 (its Appendix D, issue 48), as the Project Lead decided on 17 October 2026.* Section 9.1 has a light or a pump fade or ramp up to a `failSafeState` that lights or runs it after power-up, but one that entered such a state on loss of contact, a takeover, or a release (Sections 7.3 and 7.5) changed to it at once. So a pump that its controller stopped for a while, as for feeding, stepped from a stop to its fail-safe speed when its controller failed, and a light that its controller drove dark at night stepped to its fail-safe levels. Section 8.1.6 now says that a light or a pump that enters such a state at any other time reaches it evenly, from what it shows or runs at that moment, over the times that Device Classes Section 8.3.6 defines, whether that is up or down, and that it still reaches its inactive state, or any attribute of it, at once (Section 8.2). The ramp delays a pump's full flow by 30 s at most, which is short beside a `failSafeTimeout` of 30 s or more, and Table 3 recommends `continue` for lights, so little else changes. The test "Running fail-safe states on loss of contact" checks it.

The following issues record the Project Lead's decisions of 17 October 2026, which revision 0.4 applies:

25. **Paths that rely on a long fail-safe timeout.** *Resolved in revision 0.4, as the Project Lead decided on 17 October 2026.* Section 13.4.3 counted an output on a radio link as having a path when it stops by itself on loss of contact, whatever its `failSafeTimeout`, which may be 3600 s, so a heater on a Wi-Fi outlet could heat for an hour after its link was jammed, with no warning. It now counts such an output only while its `failSafeTimeout` is at most 300 s (provisional; issue 5), and the controller otherwise shows the standing warning. Section 13.4.4 now says that a controller's own link to its network counts as radio, as Section 5.2 already did, so that a network device is connected by wire only while the controller is too, and controllers are advised to use Ethernet. It also says that a bridged device whose `failSafe` is `device` is judged by Section 13.4.3 once the Device Protocol requires its fail-safe attributes to match its equipment, and until then by the Device Protocol's stricter rule. The test "Safety functions" checks each.

26. **Stops with no known end after a controller fails.** *Resolved in revision 0.4, with Device Classes revision 0.5, as the Project Lead decided on 17 October 2026.* A stop with no known end, such as an interlock's while a leak sensor is wet, left an output to its fail-safe behavior when its controller failed, so a return pump with the recommendation of Table 3, `run`, pumped water onto the floor. Device Classes revision 0.5 lets a controller latch a stop (its Section 7.4.13), and Section 8.4 has an endpoint keep a latched stop, in place of its fail-safe behavior and its `powerOnBehavior`, through loss of contact and power cuts, until a controller ends it with a command that changes its output. Only stops latch, so the exception never starts anything, and the Automation specification says which stops a controller latches. The test "Latched stops" checks it with a return pump stopped for a leak. Raised by the [Automation](#) specification (Appendix D, issue 17).
27. **Fans.** *Resolved in revision 0.4, with Data Model revision 0.4, as the Project Lead decided on 17 October 2026.* The Data Model adds the equipment category `fan`, and Table 3 recommends for it `run` on loss of contact, `fail-safe` after power-up, a power-on delay of 0 s, and `stop` on battery, noting that a fan that switches itself on and off by temperature can use `continue`. A cooling fan left running cools a tank by a few degrees at most, which its heater can make up, while a fan stopped in a hot room lets the tank overheat. Raised by the [Automation](#) specification (Appendix D, issue 4) and the [Data Model](#) specification (Appendix C, issue 28).
28. **What `continue` keeps for outlets with a backup thermostat and for top-off units.** *Resolved in revision 0.4, with Device Classes revision 0.5, as the Project Lead decided on 17 October 2026; the recommended defaults are unchanged.* An outlet whose backup thermostat was enabled with a setpoint regulated whenever it followed `continue`, so a heater outlet that a low-water interlock had stopped switched on in a drained sump as soon as its controller was lost. Device Classes revision 0.5 makes the backup thermostat's regulation the outlet's program, which a `set`, such as an interlock's stop, takes the outlet off, as it takes a light or a pump off its own (its Section 7.13.9); Section 7.3 and the notes of Table 3 for heaters and chillers now say so. A top-off unit whose water carries an additive, such as kalkwasser, dropped the additive's limits with `continue`, because only its controller enforces them, so the note of Table 3 for `auto-top-off` now recommends `failSafeAction` state with `failSafeState {"enabled": false}` for such a unit.