



OPEN AQUARIUM PROJECT SPECIFICATION

Controller API

PRE-DRAFT

Revision	0.1
Status	Pre-Draft. A candidate document submitted for consideration.
Date	2026-10-18
Identifier	controller-api
Authors	Trent Scholl, Project Lead, Open Aquarium Project
Editors	Trent Scholl
License	Community Specification License 1.0

Pre-Draft. This is a proposed draft. The Working Group has not approved it, and it will change. Comment by opening an issue with the `controller-api` 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 method definitions, JSON Schema, and example files published with this Specification are licensed under the Apache License 2.0 (`LICENSES/Apache-2.0.txt`).

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

Notes:

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

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

2 Compliance with Open Aquarium Tenets

2.1 Openness

The Specification is published under the Community Specification License 1.0, and its method definitions, schema, and examples under the Apache License 2.0. It adds no transport and no format of its own: its methods run in the sessions of the Device Protocol, JSON-RPC 2.0 over WebSocket and TLS 1.3, and carry the documents that the other Open Aquarium specifications define, and Web Push, which is an open standard. No membership, certification, fee, or maker’s service stands between a developer and an app that works with every conforming controller. The method definitions, a schema, a validator, and example message exchanges are published with it, so anyone can check an implementation.

2.2 Interoperability

Any conforming app works with any conforming controller, whoever made them. Every controller serves the same methods, so an owner can change their app without changing their controller, and their controller without losing their app. A Matter bridge or a home automation service on another host works with every controller through the same methods [OAP-MM Section 7.1.2]. The methods carry the configuration documents, alarm settings, alarm messages, audit records, and Data Model Documents of the other specifications unchanged, so nothing is translated, and nothing is lost on the way.

2.3 Local Control

Everything in this Specification works on the local network, with no internet connection, account, or maker’s service. Through a relay, the same methods work remotely, with the same roles, and everything that can be done remotely can also be done locally [OAP-SA Section 13.1.1]. An owner takes all their data out in one export file, at no charge, and imports it into another controller (Section 14). Firmware updates work with no internet

connection, from an image that the app gives the controller (Section 17.2), and nothing leaves the owner's network without an owner's consent (Section 19).

2.4 Safety

No method weakens a protection. Every method acts through the controller's own model: overrides, modes, scenes, step programs, and `stop-all` take the precedence that the Automation specification gives them, and interlocks, emergency modes, latched stops, fail-safe settings, and the devices' own limits still apply. Only the controller commands its devices: it refuses an action, such as a manual dose, on an output that a layer holds stopped (Section 15), changes a fail-safe setting only with the person's confirmation (Section 8.6), and holds a probe's inputs while it is calibrated until a person confirms that it is back (Section 16.3). Every temporary state that a method starts has an end: an override has a duration, a scene lasts at most 4 hours, and `stop-all` at most 24 hours. `stop-all` works in emergency sessions and while the controller's clock is not valid (Section 10.8), and no other request delays its stops. Losing an app, a session, or a bridge never affects control, which runs on the controller. Apps confirm each notification of an alarm that reaches them over a session, so that a controller knows when to use push instead (Section 11.2). A configuration document is checked before it replaces the running one, and the owner sees every warning that the controller would give about it, such as a safety function that depends only on a radio link (Section 10.1). Reading the state of devices sends nothing to them, so that apps never slow a stop (Section 8.3). Firmware is never installed in the middle of a dose, a fill, or a water change, and every image is checked against its signed manifest first (Section 17).

2.5 Sustainability

An owner who replaces a controller moves its configuration, alarm settings, and history to the new one in one export file, rather than setting everything up again. Apps outlive the controllers they were written for, and controllers outlive the apps that came with them. Owners update the firmware of the controller and of every device and module from one place, without the internet if they need to, so that equipment keeps receiving its fixes (Section 17). The methods fit the microcontroller of the reference controller: every document travels in one message, and only export files and firmware images are sent in parts.

Revision History

Revision	Date	Description	Authors
0.1	2026-10-18	Initial Pre-Draft.	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	3
2.5	Sustainability	3
	Revision History	3
3	Scope	6
4	References	6
4.1	Normative References	6
4.2	Informative References	7
5	Terms, Definitions, and Conventions	7
5.1	Requirements Language	7
5.2	Terms and Definitions	8
5.3	Notation	8
6	Introduction	8
6.1	Purpose	8
6.2	Design Principles	8
6.3	Overview	9
6.4	Example	10
7	Common Rules	10
7.1	Sessions	10
7.2	Access	10
7.3	Audit Records	11
7.4	Errors	11
7.5	Message Sizes	13
7.6	Documents, Versions, and Checks	13
7.7	Subscriptions	14
7.8	Times, Durations, and Identifiers	14
7.9	Methods	14
7.10	Extensions	20
7.11	Requests That Reach Devices	20
7.12	Long Operations	20
8	The System	21
8.1	The Controller	21
8.2	Devices and Modules	21
8.3	State	22
8.4	Identifying	23
8.5	Adding and Removing Devices	23
8.6	Device Settings	25
8.7	Restarting and Factory Reset	26
9	History	27
9.1	Reading History	27
9.2	Pages	27
10	Automation	28
10.1	The Configuration Document	28
10.2	Saved State and Status	29
10.3	Modes	30
10.4	Scenes	31
10.5	Overrides	32
10.6	Step Programs	33
10.7	Latches	33
10.8	Stopping Everything	34
10.9	Matter Bridges on Other Hosts	35
11	Alarms	35
11.1	Listing and Following Alarms	35

11.2	Notifications Through Apps	36
11.3	Acknowledging and Silencing	37
11.4	Tests	37
11.5	Push	38
11.6	Alarm Settings	38
12	Users and Sharing	40
12.1	Users	40
12.2	Invitations	40
12.3	Changing and Removing Users	40
12.4	The Audit Log	41
12.5	Passwords and Labels	42
13	Records	42
13.1	Maintenance	42
13.2	Manual Results	44
13.3	Laboratory Results	45
14	Export and Import	45
14.1	Export	45
14.2	Import	46
15	Actions	48
15.1	Actions on Demand	48
16	Calibrations	50
16.1	Kinds of Calibration	50
16.2	Following Calibrations	52
16.3	Probes	53
16.4	Dosing Heads	54
16.5	Analyzers	54
16.6	Corrections of Analyzers' Results	54
17	Firmware Updates	56
17.1	Firmware and Releases	56
17.2	Installing	57
17.3	Settings and Reverting	58
18	Remote and Emergency Access	59
18.1	Remote Access	59
18.2	Emergency Access	60
19	Privacy	61
19.1	Consent	61
19.2	Erasing Data	62
20	Conformance	62
20.1	Controllers	62
20.2	Apps and Services	62
20.3	Machine-Readable Definitions and Tools	62
20.4	Test Procedures	63
21	Security and Privacy Considerations	70
Appendix A — Approval Checklist		70
Appendix B — Message Exchanges (Informative)		71
Appendix C — Open Issues (Informative)		72

3 Scope

This Specification defines the methods that apps and services use to monitor and control a controller, covering:

- the system: the controller's identity, the revisions of the specifications it implements, and its devices and modules, with their endpoints, descriptions, and state;
- history: measurements, observations, and events for a time range, as Data Model Documents;
- automation: reading, checking, and replacing the configuration document; reading the saved state; starting and ending modes and scenes; setting and ending overrides; starting and stopping step programs; clearing latches; and `stop-all`;
- alarms: listing and following them, delivering notifications to apps, acknowledging, silencing, tests, push subscriptions, the controller's VAPID public key, and reading, checking, and replacing alarm settings;
- users and sharing: listing users, changing their roles, removing users and credentials, passwords and the labels of credentials, and reading the audit log;
- records: maintenance records and test results, including laboratory documents;
- export and import of the export file of the Data Model;
- actions on demand: feeding, tests, manual fills, manual doses, and primes;
- calibrations of probes, dosing heads, and analyzers, and corrections of analyzers' results;
- device settings and fail-safe settings, adding, removing, and identifying devices, restarts, and factory reset;
- firmware updates of the controller and of its devices and modules;
- remote access, emergency access, consent, and erasing data; and
- conformance requirements and test procedures.

This Specification does not define:

- discovery, sessions, pairing, invitations, and relays, which the Device Protocol defines, and on which this Specification builds [OAP-DP];
- users, roles, operations, and the audit log, which the Security and Access specification defines [OAP-SA];
- the documents it carries, and what a controller does with them, which the Data Model, Automation, Alarms and Notifications, and other Open Aquarium specifications define;
- the methods that devices serve, which the Device Protocol defines;
- how apps present anything to people;
- network settings, which the Device Protocol's `network.read` and `network.configure` carry for controllers as for devices [OAP-DP Section 13.3]; or
- turning on a Matter bridge, opening its commissioning window, and removing its fabrics [OAP-MM Sections 7.2 and 7.3], allowing firmware that a manufacturer did not sign, which needs physical presence [OAP-SA Section 15.8], turning owner recovery off [OAP-SA Section 11.7.3], and signing out a user's other sessions, which a controller offers through means of its own, such as its own buttons, its web interface, or extension methods (Section 7.10).

4 References

4.1 Normative References

- **[RFC 2119]** Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- **[RFC 8174]** Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, May 2017.
- **[RFC 8259]** Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, December 2017.
- **[JSON-RPC]** JSON-RPC Working Group, "JSON-RPC 2.0 Specification", 2013-01-04, <https://www.jsonrpc.org/specification>.
- **[RFC 3339]** Klyne, G. and C. Newman, "Date and Time on the Internet: Timestamps", RFC 3339, July 2002.
- **[RFC 4648]** Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, October 2006.

- **[RFC 6901]** Bryan, P., Ed., Zyp, K., and M. Nottingham, Ed., “JavaScript Object Notation (JSON) Pointer”, RFC 6901, April 2013.
- **[RFC 9562]** Davis, K., Peabody, B., and P. Leach, “Universally Unique IDentifiers (UUIDs)”, RFC 9562, May 2024.
- **[FIPS 180-4]** National Institute of Standards and Technology, “Secure Hash Standard (SHS)”, FIPS PUB 180-4, August 2015.
- **[RFC 8030]** Thomson, M., Damaggio, E., and B. Raymor, Ed., “Generic Event Delivery Using HTTP Push”, RFC 8030, December 2016.
- **[RFC 8291]** Thomson, M., “Message Encryption for Web Push”, RFC 8291, November 2017.
- **[RFC 8292]** Thomson, M. and P. Beverloo, “Voluntary Application Server Identification (VAPID) for Web Push”, RFC 8292, November 2017.
- **[JSON Schema]** “JSON Schema: A Media Type for Describing JSON Documents” and “JSON Schema Validation”, draft 2020-12, <https://json-schema.org/draft/2020-12>.
- **[OAP-DM]** Open Aquarium Project, “Open Aquarium Data Model”, Revision 0.4, [specs/data-model/spec.md](#).
- **[OAP-DC]** Open Aquarium Project, “Open Aquarium Device Classes”, Revision 0.5, [specs/device-classes/spec.md](#).
- **[OAP-FS]** Open Aquarium Project, “Open Aquarium Fail-Safe Behavior”, Revision 0.4, [specs/fail-safe-behavior/spec.md](#).
- **[OAP-DP]** Open Aquarium Project, “Open Aquarium Device Protocol”, Revision 0.3, [specs/device-protocol/spec.md](#).
- **[OAP-SA]** Open Aquarium Project, “Open Aquarium Security and Access”, Revision 0.3, [specs/security-and-access/spec.md](#).
- **[OAP-AU]** Open Aquarium Project, “Open Aquarium Automation”, Revision 0.3, [specs/automation/spec.md](#).
- **[OAP-AN]** Open Aquarium Project, “Open Aquarium Alarms and Notifications”, Revision 0.3, [specs/alarms-and-notifications/spec.md](#).
- **[OAP-LP]** Open Aquarium Project, “Open Aquarium Lighting Programs”, Revision 0.3, [specs/lighting-programs/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](#).
- **[OAP-DO]** Open Aquarium Project, “Open Aquarium Dosing”, Revision 0.3, [specs/dosing/spec.md](#).
- **[OAP-WT]** Open Aquarium Project, “Open Aquarium Water Testing”, Revision 0.3, [specs/water-testing/spec.md](#).
- **[OAP-FP]** Open Aquarium Project, “Open Aquarium Flow Programs”, Revision 0.3, [specs/flow-programs/spec.md](#).
- **[OAP-PI]** Open Aquarium Project, “Open Aquarium Probe Interface”, Revision 0.4, [specs/probe-interface/spec.md](#).
- **[OAP-PM]** Open Aquarium Project, “Open Aquarium Power Module”, Revision 0.3, [specs/power-module/spec.md](#).
- **[OAP-MB]** Open Aquarium Project, “Open Aquarium Module Bus”, Revision 0.4, [specs/module-bus/spec.md](#).
- **[OAP-MM]** Open Aquarium Project, “Open Aquarium Matter Mapping”, Revision 0.3, [specs/matter-mapping/spec.md](#).

4.2 Informative References

- **[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 Protocol [OAP-DP], such as party, app, service, unit, session, message, method, request, response, and notification; in the Security and Access specification [OAP-SA], such as user, credential, role, operation, owner, emergency session, and audit log; in the Automation specification [OAP-AU], such as configuration document, element, output, mode, override, step program, latch, interlock, saved state, and `stop-all`; in the Alarms and Notifications specification [OAP-AN], such as alarm, notice, alarm message, recipient, channel, feed, and alarm settings document; and in the Data Model [OAP-DM], such as Document and export file, have the same meaning here.

Namespace: The part of a method's name before the dot, such as `automation` in `automation.startMode`.

Session's user: The user whose credential authenticated a session, or who signed in to it with a password [OAP-DP Section 10.7].

Version: A string that a controller gives a document that it holds, and changes whenever the document changes (Section 7.6).

Check: A request that works out what a change would do, and reports it, without changing anything: `automation.validateConfiguration`, `alarms.validateSettings`, and a request with `check true`.

Output status: What decides an output now: its resolved state, the layer of Automation Table 25 that decided it, the element of that layer, and when it ends (Section 10.2).

Matter bridge's user: The user of kind `service` that a controller created for a Matter bridge when an owner turned the bridge on [OAP-MM Section 7.2.3].

File: Bytes that a controller holds for one session: an export file that it made for the session, or one that the session uploads to it (Section 14).

5.3 Notation

Method names, JSON members, and the names of operations, attributes, and codes appear in `monospace`. Durations are in seconds, and sizes in bytes, unless stated. In examples, the `jsonrpc` member that every message carries is left out, and whitespace is added. References to other Open Aquarium specifications name them, as in "Automation Section 15", or use their tags from Section 4, as in "[OAP-AU Section 15]". A section number with a third part refers to a numbered item of that section: "Section 10.5.2" means item 2 of Section 10.5.

6 Introduction

6.1 Purpose

A controller runs an aquarium, and apps show it to people and let them operate it. The Device Protocol lets an app find a controller, pair with it, and open an authenticated session, but its methods are those that devices need. Without common methods for the rest, each controller's app would speak its own language: an owner who changed one would lose the other, and a Matter bridge or a home automation service on another host could not work with every controller. This Specification defines those methods, and only those that the other Open Aquarium specifications already need apps to use.

6.2 Design Principles

1. **Built on the Device Protocol.** The methods run in its sessions, with its roles, errors, and relays; there is no new transport.
2. **The other specifications' documents, unchanged.** Configuration documents, alarm settings, alarm messages, audit records, and Data Model Documents travel as their own specifications define them.
3. **Operations as Security and Access defines them.** Every method names the operation it needs, and every audited method says what the audit log records for it.

4. **The controller decides.** Methods ask the controller to do what its own model allows, such as an override, a mode, or a manual dose; the controller sends every command itself, as the device's controller, and none passes an interlock.
5. **Check before change.** A document is checked before it replaces the running one, and the owner sees what the controller would warn of.
6. **No lost changes.** Documents have versions, so that two apps never undo each other's changes unknowingly, and a bridge never replaces a person's override.
7. **Fits a microcontroller.** Every document travels in one message; only export files are sent in parts.

6.3 Overview

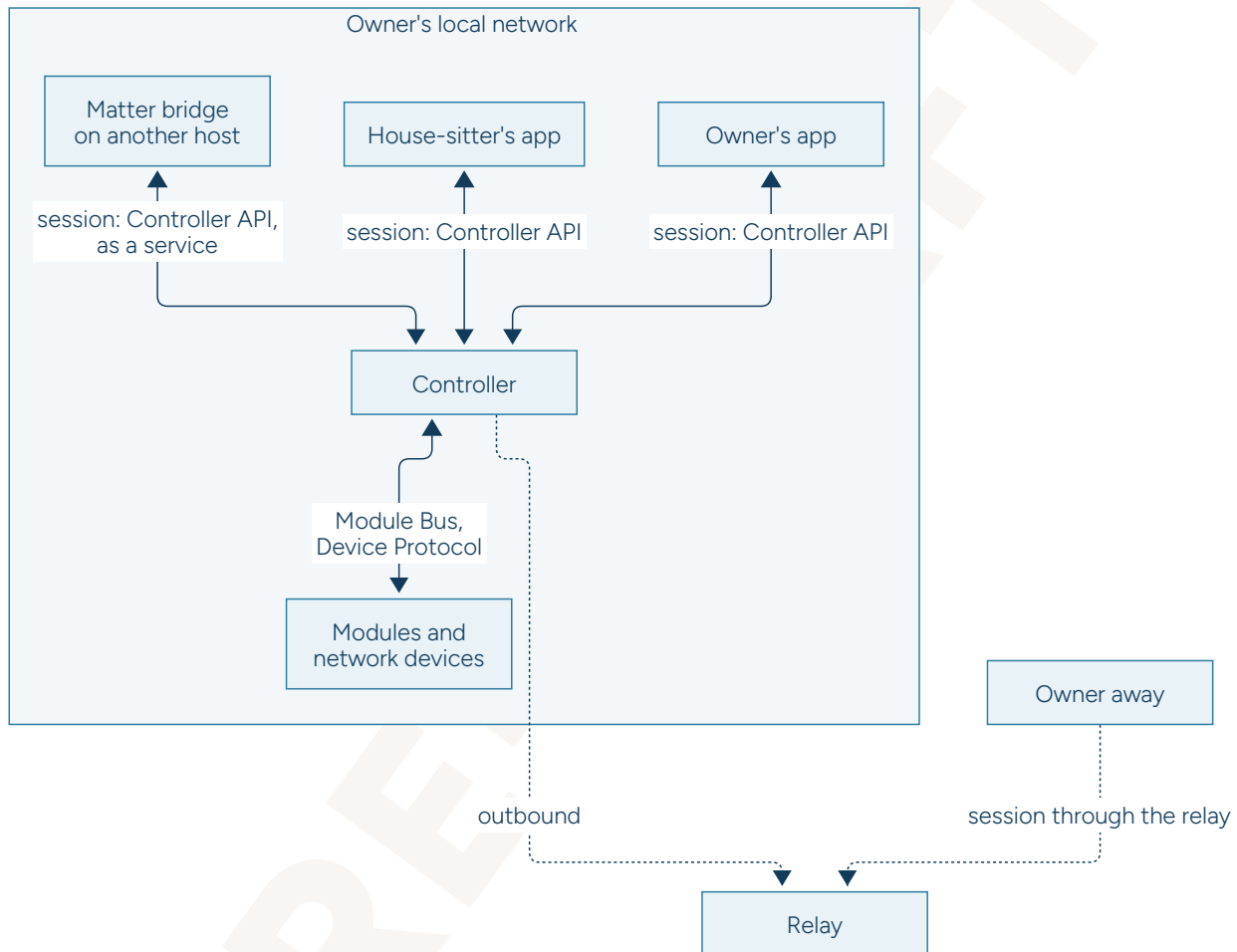


Figure 1 — Apps, services, and the controller

Apps and services open sessions with a controller, as the Device Protocol defines, and call the methods of this Specification in them. The controller checks each request against the role of the session's user, carries it out through its own model, and records it in its audit log. It reports changes on the sessions that asked for them, and delivers notifications of alarms to the apps of the people they are for. Only the controller talks to its devices.

The methods are grouped in namespaces, as Table 1 shows.

Table 1 — Namespaces

Namespace	Section	What its methods cover
system	8	The controller, its devices and modules, their state and settings, adding and removing them, restarts, and factory reset.
history	9	Measurements, observations, and events, as Data Model Documents.

Namespace	Section	What its methods cover
automation	10	The configuration document, the saved state, modes, scenes, overrides, step programs, latches, and <code>stop-all</code> .
alarms	11	Alarms, their notifications to apps, push, and the alarm settings.
users	12	Users, their passwords and credentials, and the audit log.
records	13	Maintenance records and test results.
export	14	Export files.
actions	15	Feeding, tests, fills, doses, and primes, on demand.
calibration	16	Calibrations of probes, dosing heads, and analyzers, and corrections of analyzers' results.
updates	17	Firmware updates of the controller and of its devices and modules.
access	18	Remote access and emergency access.
privacy	19	Consent, and erasing data.

6.4 Example

An owner sees a temperature alarm on their phone, acknowledges it, and pauses the pumps to feed the fish. The app has a session with the controller, in which it subscribed to alarms:

```
{ "id": 21, "method": "alarms.acknowledge", "params": { "alarms": ["6f1c0e2a-4b7d-4e19-9a53-0d2c8b7e4f61"] } }
{ "id": 21, "result": { "acknowledged": ["6f1c0e2a-4b7d-4e19-9a53-0d2c8b7e4f61"] } }
{ "id": 22, "method": "automation.startMode", "params": { "mode": "feed-a" } }
{ "id": 22, "result": { "left": 600, "until": "2026-10-18T18:10:00+10:00" } }
```

The controller records both in its audit log, with the owner as the actor, stops the pumps that the feed mode lists, with a `revertAfter` of 600 s so that they restart even if the controller fails [OAP-AU Section 17.5], and tells every app that subscribed to the `automation` namespace.

7 Common Rules

7.1 Sessions

1. Apps and services call the methods of this Specification in sessions with a controller, as the Device Protocol defines them: over TLS 1.3 with both parties authenticated, on the local network or through a relay [OAP-DP Sections 9 and 15.4]. There is no other transport.
2. A controller MUST serve every request of Table 3 that an app or service sends it, on every session whose party it trusts [OAP-DP Section 10.1].
3. The controller carries out each request for the session's user, and checks it against that user's role (Section 7.2).
4. The namespaces of Table 1 are not those of any method of the Device Protocol, so a controller serves the methods of both specifications on the same session, and the Device Protocol's rules for messages, requests, responses, and errors apply to the methods of this Specification [OAP-DP Sections 9.3 to 9.6].
5. A party that is not a user, such as a device, has no role, and is permitted no operation of this Specification.

7.2 Access

1. Each method of Table 3 names the operation of Security and Access Table 6 that it needs. Before it carries out a request, a controller MUST check that the role of the session's user permits that operation, at the time of the request, and MUST refuse the request with `not-permitted` otherwise [OAP-SA Section 11.3.1]. A change of a user's role therefore applies at once to the sessions that the user has open.
2. Where Table 3 names more than one operation, the method's section says which one a request needs, and a request that needs several needs each of them.
3. In an emergency session, a controller MUST refuse every request whose operation the `override` role does not permit, and every request that needs `manage-own-credentials`, with `not-permitted` and the reason `emergency-session` [OAP-SA Section 14.5].

- While its clock is not valid, a controller MUST refuse, with `not-permitted` and the reason `grant-unverified`, every request of a user whose grant has an expiry, except requests that need `view-state`, `automation.stopAll`, `automation.endStopAll`, and `automation.setOverride` with the state `stop` [OAP-SA Section 10.7.2]. So a house-sitter can still stop equipment after a power cut, but cannot do what an expired grant would forbid.
- Reading needs no more than the `view` role, except the users, which need `view-access`, and the audit log, which needs `view-audit-log` [OAP-SA Table 6].

7.3 Audit Records

- For each request of a method whose entry in Table 3 is audited, a controller MUST record an audit record [OAP-SA Section 12.2], whether it succeeds, fails, or is refused, with these members:
 - `action`, `target`, and `details`, as the method's section says;
 - `actor`: the session's user, with its `kind`, `user`, `name`, and `role`, and, for a session that a credential authenticated, that `credential`;
 - `via`: `matter` for a request with `fabric` (Section 10.9), `emergency` in an emergency session, `remote` in a session through a relay, and `local` otherwise; and
 - `result`: `success`; `denied` for a request refused with `not-permitted`; and `failed` for any other error.
- A controller MUST NOT record in an audit record a password, a webhook secret, a push subscription's `p256dh` or `auth`, a code, or a key [OAP-SA Section 12.3.4].
- Reads and checks are not recorded, because they change nothing [OAP-SA Section 12.1.4]. `export.create` records an export, and `export.import` an import, so that their parts are not recorded one by one (Section 14).

7.4 Errors

- A controller refuses a request with the error codes of the Device Protocol, and with no other code, as that Specification requires [OAP-DP Section 9.6].
- A controller MUST refuse with `invalid-params` a request whose parameters are not valid against the method's definition in the schema (Section 20.3).
- When it refuses a request for one of the reasons of Table 2, a controller MUST send the error that Table 2 gives, with `data` that has the reason as `reason`, and with the members that the reason names, such as `errors`, the problems of a document (Section 7.6.4). An error's `data` for a method of this Specification MUST be valid against `$defs/errorData` of the schema.
- A controller MUST NOT carry out any part of a request that it refuses.

The reasons of Table 2 tell an app why its request was refused, so that it can tell the person, without parsing the error's `message`. Each method's section lists the reasons it may give, besides `emergency-session` and `grant-unverified`, which any method may give.

Table 2 — Reasons

Reason	Error	Meaning
<code>invalid-document</code>	<code>bad-parameter</code>	A document is not valid; <code>errors</code> lists the problems.
<code>version-conflict</code>	<code>bad-state</code>	The document has changed since the version given in <code>ifVersion</code> .
<code>unknown-element</code>	<code>bad-parameter</code>	The configuration has no element of that identifier and kind, such as no mode, output, program, latch, or lighting function, or no system or equipment of that identifier.
<code>unknown-device</code>	<code>bad-endpoint</code>	The controller has no such device, or the device no such endpoint.
<code>unknown-attribute</code>	<code>bad-attribute</code>	The endpoint, or the device, has no such attribute.
<code>unknown-cursor</code>	<code>bad-parameter</code>	The cursor is not one the controller gave, or it has expired.
<code>unknown-alarm</code>	<code>bad-parameter</code>	No alarm or notice that is not closed has that <code>id</code> .
<code>unknown-recipient</code>	<code>bad-parameter</code>	The alarm settings have no such recipient, or no such feed.
<code>unknown-user</code>	<code>bad-parameter</code>	The controller has no such user.
<code>unknown-credential</code>	<code>bad-parameter</code>	No user has that credential.
<code>unknown-file</code>	<code>bad-parameter</code>	No file of that name is offered to, or being uploaded on, the session.

Reason	Error	Meaning
invalid-state	bad-parameter	The state is not one the output can take, such as a level for an output whose class has none.
duration-too-long	bad-parameter	The duration is longer than the scene's own.
start-passed	bad-parameter	The start time has passed.
action-needed	bad-parameter	Clearing an aborted water change needs <code>action</code> .
service-owner	bad-parameter	A service cannot have the <code>owner</code> role.
owner-expiry	bad-parameter	A grant with the <code>owner</code> role cannot expire.
invalid-record	bad-parameter	The members given do not fit the kind of record.
invalid-result	bad-parameter	A result's unit is not accepted for its parameter, or its system or vessel is not the controller's.
not-laboratory	bad-parameter	An observation of the document does not belong to a sample of a laboratory.
invalid-export	bad-parameter	The file is not a valid export file; <code>errors</code> lists the problems.
file-too-large	bad-parameter	The file is larger than the controller can hold; <code>limit</code> gives the largest, in bytes.
out-of-order	bad-parameter	The part's offset is not the number of bytes received so far.
not-by-person	bad-state	A person cannot start or end a power-loss mode (Automation Section 14.6.3).
unsupported	bad-state	The element is of a type that the controller does not support (Automation Section 19.2.3).
other-user	bad-state	With <code>onlyOwn</code> , another user set the override or started the mode.
program-conflict	bad-state	The program runs already, or another program that runs sets one of its functions or outputs (Automation Section 16.5).
clock-invalid	bad-state	The request needs a valid clock.
latch-set	bad-state	The latch's <code>set</code> condition is true or unknown, so it cannot be cleared (Automation Section 10.4.2).
owner-reset-off	bad-state	The latch's <code>ownerReset</code> is <code>false</code> .
not-latched	bad-state	There is no aborted water change to resume or abandon.
cannot-resume	bad-state	The water change cannot resume: a fill input is unavailable, or there is a leak (Water Level and Water Change Section 9.11.2).
device-lost	bad-state	The controller has lost the device.
device-refused	bad-state	The device refused the command; <code>error</code> holds its error.
last-owner	bad-state	The change would leave the controller without an owner (Security and Access Section 11.1.4).
no-credential	bad-state	The session was opened by signing in with a password, so it has no credential.
incomplete	bad-state	The upload is not complete, or its SHA-256 hash does not match.
already-added	bad-state	The device is in the system already.
not-found	bad-state	No unit with a pairing window open, and with the code's discriminator or the key ID given, answered; it comes in <code>system.deviceAdded</code> .
confirmation-needed	bad-parameter	The change of fail-safe settings needs the person's confirmation, and <code>confirmed</code> is not <code>true</code> .
protected-setting	bad-parameter	The controller does not make this change, such as <code>continue</code> for a return pump, or an attribute that it writes from the configuration.
wrong-class	bad-command	The endpoint's class does not have the command that the method sends.
held	bad-state	A layer of Automation Table 25 holds the endpoint's output stopped, or another hold of its function's specification holds it.
under-local-control	local-control	The endpoint is under local control (Fail-Safe Behavior Section 12).
limit-reached	bad-state	A limit that the controller enforces leaves no room, such as the day's fill time or priming time.
too-long	bad-parameter	The duration is longer than the endpoint allows, such as its <code>maxContinuousRun</code> ; <code>limit</code> gives the longest, in seconds.
no-calibration	bad-state	No calibration, or no calibration hold that waits for that confirmation, is in progress on the endpoint.
reference-needed	bad-parameter	The controller cannot work out the reference's value from what was given.

Reason	Error	Meaning
invalid-reference	bad-parameter	The result cannot be a reference, such as one that is not confirmed, or of a parameter that no analyzer measures.
no-proposal	bad-state	There is no proposed correction, and the request gives none of its own.
manifest-refused	bad-parameter	The manifest or the image failed the checks of Security and Access Section 15.6.6; errors lists why.
operation-running	bad-state	An endpoint is dosing, filling, or taking part in a water change (Fail-Safe Behavior Section 11.2.2).
update-running	busy	Another firmware update of the unit is in progress.
no-previous	bad-state	The unit has no previous firmware that its security version allows.
no-consent	bad-state	The request needs an owner's consent to update checks (Security and Access Section 15.5.1).
weak-password	bad-parameter	The password is shorter than 15 characters, or on the list of common and compromised passwords (Security and Access Section 10.5.1).
unknown-purpose	bad-parameter	The unit has no such purpose.
unknown-slot	bad-parameter	No emergency code has that slot.
device-unanswered	bad-state	The device did not answer in time, so the controller does not know whether it carried out the command (Section 7.11).
not-bridge	not-permitted	fabric is accepted only from a Matter bridge's user.
emergency-session	not-permitted	An emergency session allows only the operations of the override role (Security and Access Section 14.5).
grant-unverified	not-permitted	The user's grant has an expiry, and the controller's clock is not valid (Security and Access Section 10.7.2).

7.5 Message Sizes

1. A controller MUST accept messages of at least 1 048 576 bytes (provisional), and say so in the `maxMessageSize` of its `session.hello` [OAP-DP Section 9.3.6].
2. An app or service that reads or replaces a configuration document or an alarm settings document MUST accept messages of at least 1 048 576 bytes, so that each document arrives in one message.
3. A controller MUST NOT accept a configuration document, or an alarm settings document, whose UTF-8 encoding is longer than 1 000 000 bytes (provisional), so that each fits in one message with room to spare; it refuses a longer one as not valid, saying why in `errors` [OAP-AU Section 21.7].
4. A controller MUST make each page of history (Section 9.2), and each response of `users.readAudit`, small enough to fit in the largest message that the app accepts, by returning fewer records rather than the error `too-large`.

7.6 Documents, Versions, and Checks

1. A controller MUST give its configuration document and its alarm settings document each a version: a string of 1 to 64 characters that changes whenever the document changes, by any means, and stays the same otherwise, across restarts too.
2. When a request that replaces a document has `ifVersion`, and the document's version is not `ifVersion`, a controller MUST refuse the request with `bad-state` and the reason `version-conflict`. So two apps that edit the same document never undo each other's changes unknowingly: the second learns that the document changed, reads it again, and applies its change to the new one.
3. A check MUST report what the request would do without it, and MUST NOT change anything.
4. Each problem in `errors` has a `message` for people, and, where it applies, the `path` of the member, as a JSON Pointer [RFC 6901] into the document, such as `/functions/3/setpoint`.
5. Each warning has a `code`, the `element` it concerns where there is one, and a `message` for people. Its `code` is the code of the fault or condition that the document would raise, such as `unsupported`, `binding-missing`, or `safety-path-missing` [OAP-AU Table 27], or `warning` for any other warning that a specification requires a controller to give about a document, such as the warning of Device Protocol Section 16.4.7. A check and a replacement MUST return every warning that the specifications require the controller to give the owner about the document.

- Before it uses a configuration document or alarm settings that came from elsewhere, a controller shows the owner what they will do, and has the owner confirm them [OAP-AU Section 21.2; OAP-AN Section 19.5.3]. Through this Specification, it does so through the app: an app **MUST NOT** send a request that replaces a document with one that came from elsewhere, such as a file, a shared configuration, or another controller, until it has shown the person what those sections require, from the result of a check, and the person has confirmed it.

7.7 Subscriptions

- A session has at most one subscription of each of the `system`, `automation`, `alarms`, and `calibration` namespaces. Subscribing again replaces it, and the namespace's `unsubscribe`, or the end of the session, ends it.
- A controller **MUST** send a namespace's notifications on a session only while the session has a subscription of that namespace, from when it responds to the subscription, except those that report an operation that the session started (Section 7.12), and **MUST** send them in the order of the changes they report.
- A controller **MUST NOT** let notifications, or anything else that it does for apps, delay its control functions or its supervision of devices [OAP-SA Section 10.9.1].

Notifications get no response, so a session that ended has missed those sent meanwhile: an app that opens a new session subscribes again, and reads the current state, before it relies on them.

7.8 Times, Durations, and Identifiers

- Timestamps are those of the Device Protocol [OAP-DP Table 8], and a member that may be `null` while the controller's clock is not valid says so.
- Durations are measured as elapsed time.
- A controller gives the end of a temporary state as `left`, the seconds until it ends, and, while its clock is valid, as `until`, so that an app needs neither its own clock nor the controller's to show the time left.
- Devices are identified as Automation Section 8.5 says, elements by their Automation identifiers [OAP-AU Section 7.3], users by their `id` [OAP-SA Section 11.1], and alarms by their `id` [OAP-AN Section 12.2.3].

7.9 Methods

Table 3 lists the methods. Requests get a response, and notifications do not. `alarms.notify` is a request that the controller sends to an app (Section 11.2). Each method's section defines its parameters and its result, which `methods.json` and the schema define in machine-readable form (Section 20.3).

Table 3 — Methods

Method	Type	From	Operation	Audited	Description
<code>system.describe</code>	request	app or service	<code>view-configuration</code>	—	Returns the controller's identity, the revisions of the specifications it implements, and its clock.
<code>system.listDevices</code>	request	app or service	<code>view-configuration</code>	—	Returns the controller's devices and modules, with their endpoints.
<code>system.describeDevice</code>	request	app or service	<code>view-configuration</code>	—	Returns the description of one device or module.
<code>system.identify</code>	request	app or service	<code>identify</code>	—	Makes a device, or one of its endpoints, show where it is, without changing any output.
<code>system.read</code>	request	app or service	<code>view-state</code>	—	Returns the current values of attributes of a device or of one of its endpoints, as the controller last received them.
<code>system.subscribe</code>	request	app or service	<code>view-state</code>	—	Starts <code>system.state</code> and <code>system.devicesChanged</code> notifications on the session, replacing any earlier

Method	Type	From	Operation	Audited	Description
					subscription of the <code>system</code> namespace.
<code>system.unsubscribe</code>	request	app or service	<code>view-state</code>	—	Ends the session's subscription of the <code>system</code> namespace.
<code>system.state</code>	notification	controller	—	—	Reports the state of a device or of one of its endpoints.
<code>system.devicesChanged</code>	notification	controller	—	—	Reports that the controller's devices, or whether it has lost one, changed.
<code>system.discover</code>	request	app or service	<code>view-configuration</code>	—	Returns the units that the controller finds with a pairing window open, which an owner can add.
<code>system.addDevice</code>	request	app or service	<code>add-device</code> or <code>change-network-settings</code>	Yes	Starts adding a network device to the system: claiming it with its setup code, pairing with an invitation it made, or taking on a device that an owner's app has claimed.
<code>system.deviceAdded</code>	notification	controller	—	—	Reports the end of an operation of <code>system.addDevice</code> , to the session that started it.
<code>system.removeDevice</code>	request	app or service	<code>remove-device</code> or <code>factory-reset</code>	Yes	Removes a device or module from the system: the controller releases it, factory resets it if asked, and leaves its pairing list.
<code>system.writeSettings</code>	request	app or service	<code>change-device-settings</code> or <code>change-fail-safe-settings</code> or <code>install-firmware</code>	Yes	Changes config attributes of a device or of one of its endpoints, all or none.
<code>system.restart</code>	request	app or service	<code>restart</code>	Yes	Restarts a device, or the controller itself.
<code>system.factoryReset</code>	request	app or service	<code>factory-reset</code>	Yes	Factory resets the controller (Security and Access Section 9.8).
<code>history.read</code>	request	app or service	<code>view-history</code>	—	Returns the history of a time range as a Data Model Document, one page at a time.
<code>automation.readConfiguration</code>	request	app or service	<code>view-configuration</code>	—	Returns the automation configuration document.
<code>automation.validateConfiguration</code>	request	app or service	<code>view-configuration</code>	—	Checks a configuration document as <code>automation.replaceConfiguration</code> would, without changing anything.
<code>automation.replaceConfiguration</code>	request	app or service	<code>change-configuration</code>	Yes	Replaces the automation configuration document.
<code>automation.readState</code>	request	app or service	<code>view-state</code>	—	Returns the saved state, and what decides each output now.
<code>automation.subscribe</code>	request	app or service	<code>view-state</code>	—	Starts <code>automation.changed</code> notifications on the session.
<code>automation.unsubscribe</code>	request	app or service	<code>view-state</code>	—	Ends the session's subscription of the <code>automation</code> namespace.

Method	Type	From	Operation	Audited	Description
automation.changed	notification	controller	—	—	Reports that the saved state, the status of an output, or the configuration document changed.
automation.startMode	request	app or service	start-mode	Yes	Starts a mode, or starts an active mode again.
automation.endMode	request	app or service	start-mode or end-emergency-mode	Yes	Ends a mode.
automation.startScene	request	app or service	start-mode	Yes	Starts a scene of a lighting function.
automation.endScene	request	app or service	start-mode	Yes	Ends the scene that a person started on a lighting function.
automation.setOverride	request	app or service	set-override	Yes	Sets an override of an output, replacing any override it has.
automation.endOverride	request	app or service	set-override	Yes	Ends the override of an output.
automation.startProgram	request	app or service	start-program	Yes	Starts a step program, now or at a later time.
automation.stopProgram	request	app or service	start-program	Yes	Stops a step program.
automation.clearLatch	request	app or service	clear-latch	Yes	Resets a latch, clears a latching interlock or a latched fault, or resumes or abandons an aborted water change.
automation.stopAll	request	app or service	stop-all	Yes	Stops every output at once, as Security and Access Section 11.8 defines, or sets a new end for a stop-all in effect.
automation.endStopAll	request	app or service	stop-all	Yes	Ends stop-all.
alarms.list	request	app or service	view-state	—	Returns every alarm and notice that is not closed.
alarms.subscribe	request	app or service	view-state	—	Starts alarms.changed notifications on the session, and, unless notify is false, delivery of the app channel's messages for the session's user with alarms.notify.
alarms.unsubscribe	request	app or service	view-state	—	Ends the session's subscription of the alarms namespace.
alarms.changed	notification	controller	—	—	Reports a change of an alarm: an alarm message of the kind raised, acknowledged, silenced, cleared, or closed.
alarms.notify	request	controller	—	—	Delivers a message of the app channel to the app of one of its recipients, which responds once it has received it.
alarms.acknowledge	request	app or service	acknowledge-alarm	Yes	Acknowledges alarms (Alarms and Notifications Section 12.4).
alarms.silence	request	app or service	acknowledge-alarm	Yes	Silences the alarms that are sounding (Alarms and Notifications Section 12.5).
alarms.test	request	app or service	manage-consent or manage-own-credentials	Yes	Sends a test message to a recipient or a feed (Alarms and Notifications Section 16.8).

Method	Type	From	Operation	Audited	Description
<code>alarms.readVapidKey</code>	request	app or service	<code>view-configuration</code>	—	Returns the controller's VAPID public key, with which apps subscribe to push (Alarms and Notifications Section 17.2.2).
<code>alarms.registerPush</code>	request	app or service	<code>manage-own-credentials</code>	Yes	Registers the push subscription of the app installation whose credential opened the session, replacing any it had.
<code>alarms.unregisterPush</code>	request	app or service	<code>manage-own-credentials</code>	Yes	Removes the push subscription of the app installation whose credential opened the session.
<code>alarms.readSettings</code>	request	app or service	<code>view-configuration</code>	—	Returns the alarm settings document, without its secrets.
<code>alarms.validateSettings</code>	request	app or service	<code>view-configuration</code>	—	Checks an alarm settings document as <code>alarms.replaceSettings</code> would, without changing anything.
<code>alarms.replaceSettings</code>	request	app or service	<code>change-configuration</code> or <code>manage-consent</code> or <code>manage-own-credentials</code>	Yes	Replaces the alarm settings document.
<code>users.list</code>	request	app or service	<code>view-access</code>	—	Returns the controller's users.
<code>users.update</code>	request	app or service	<code>manage-users</code>	Yes	Changes a user's name, role, expiry, or remote access.
<code>users.remove</code>	request	app or service	<code>manage-users</code>	Yes	Removes a user, and revokes every credential of the user.
<code>users.revokeCredential</code>	request	app or service	<code>manage-users</code> or <code>manage-own-credentials</code>	Yes	Revokes one credential of a user, such as a lost phone's.
<code>users.readAudit</code>	request	app or service	<code>view-audit-log</code>	—	Returns records of the controller's audit log, in order.
<code>users.setPassword</code>	request	app or service	<code>manage-own-credentials</code>	Yes	Sets the password of the session's own user, for the web interface and for services that sign in (Security and Access Section 10.5).
<code>users.removePassword</code>	request	app or service	<code>manage-users</code> or <code>manage-own-credentials</code>	Yes	Removes a user's password.
<code>users.labelCredential</code>	request	app or service	<code>manage-users</code> or <code>manage-own-credentials</code>	Yes	Changes the label of a credential, such as "Alex's phone".
<code>records.recordMaintenance</code>	request	app or service	<code>record-maintenance</code>	Yes	Records maintenance that changes no output: a refill, a replaced reagent, an emptied waste container, or other maintenance.
<code>records.enterResults</code>	request	app or service	<code>enter-results</code>	Yes	Enters the results of manual tests (Water Testing Section 12.3).
<code>records.importResults</code>	request	app or service	<code>enter-results</code>	Yes	Imports the results of a laboratory, as a Data Model Document (Water Testing Section 12.4).
<code>export.create</code>	request	app or service	<code>export-data</code>	Yes	Makes an export file (Data Model Section 7.8), and offers it to the session.

Method	Type	From	Operation	Audited	Description
<code>export.read</code>	request	app or service	<code>export-data</code>	With another method	Reads part of an export file that the controller offered to the session.
<code>export.upload</code>	request	app or service	<code>change-configuration</code> or <code>enter-results</code>	With another method	Sends the controller part of an export file to import.
<code>export.import</code>	request	app or service	<code>change-configuration</code> or <code>manage-consent</code> or <code>enter-results</code>	Yes	Checks an export file that the session uploaded, or imports it.
<code>actions.feed</code>	request	app or service	<code>run-action</code>	Yes	Feeds now, with a feeder's <code>feed</code> command (Device Classes Section 9.9).
<code>actions.test</code>	request	app or service	<code>run-action</code>	Yes	Starts a test on demand, or a test of a standard, with an analyzer's <code>test</code> command (Water Testing Section 8.8).
<code>actions.fill</code>	request	app or service	<code>run-action</code> or <code>dose-now</code>	Yes	Starts a manual fill of a top-off unit, with its <code>fill</code> command (Water Level and Water Change Section 8.14).
<code>actions.dose</code>	request	app or service	<code>dose-now</code>	Yes	Gives a manual dose, forward or in reverse, with a dosing head's <code>dose</code> command (Dosing Section 10.6).
<code>actions.prime</code>	request	app or service	<code>dose-now</code>	Yes	Primes a dosing head, with its <code>prime</code> command (Dosing Section 10.6).
<code>calibration.list</code>	request	app or service	<code>view-state</code>	—	Returns every calibration in progress, and every calibration hold that waits for a person.
<code>calibration.subscribe</code>	request	app or service	<code>view-state</code>	—	Starts <code>calibration.changed</code> notifications on the session.
<code>calibration.unsubscribe</code>	request	app or service	<code>view-state</code>	—	Ends the session's subscription of the <code>calibration</code> namespace.
<code>calibration.changed</code>	notification	controller	—	—	Reports that a calibration, or a calibration hold, changed.
<code>calibration.start</code>	request	app or service	<code>calibrate</code>	Yes	Starts the calibration of a probe, a dosing head, or an analyzer (Table 14).
<code>calibration.point</code>	request	app or service	<code>calibrate</code>	Yes	Records one point of a probe's calibration (Probe Interface Section 14.7.2).
<code>calibration.complete</code>	request	app or service	<code>calibrate</code>	Yes	Completes the calibration of a probe or of a dosing head.
<code>calibration.cancel</code>	request	app or service	<code>calibrate</code>	Yes	Cancels a calibration in progress, which keeps the calibration in use.
<code>calibration.confirm</code>	request	app or service	<code>calibrate</code>	Yes	Gives a person's confirmation that a calibration hold waits for (Probe Interface Section 14.8).
<code>calibration.reset</code>	request	app or service	<code>calibrate</code>	Yes	Resets a probe's calibration, for a probe that a person replaced (Probe Interface Section 16.5.2).
<code>calibration.listCorrections</code>	request	app or service	<code>view-configuration</code>	—	Returns the corrections of analyzers' results, their comparisons with references,

Method	Type	From	Operation	Audited	Description
					and the corrections the controller proposes (Water Testing Section 11).
<code>calibration.setReference</code>	request	app or service	<code>calibrate</code>	Yes	Chooses whether a result is a reference with which analyzers' results are compared (Water Testing Section 11.2).
<code>calibration.applyCorrection</code>	request	app or service	<code>calibrate</code>	Yes	Applies a correction to an analyzer's results, as the controller proposes it or as the person gives it (Water Testing Section 11.5).
<code>calibration.dismissCorrection</code>	request	app or service	<code>calibrate</code>	Yes	Dismisses the correction that the controller proposes for an analyzer's parameter (Water Testing Section 11.4).
<code>updates.list</code>	request	app or service	<code>view-configuration</code>	—	Returns the firmware of the controller and of each of its devices and modules, and the updates available for each (Security and Access Section 15.9.2).
<code>updates.upload</code>	request	app or service	<code>install-firmware</code>	With another method	Sends the controller part of a firmware image to install, for an update without the internet (Security and Access Section 15.5.4).
<code>updates.install</code>	request	app or service	<code>install-firmware</code>	Yes	Checks an update of the controller or of a device, or starts installing it.
<code>updates.progress</code>	notification	controller	—	—	Reports the progress of an operation of <code>updates.install</code> , to the session that started it.
<code>updates.configure</code>	request	app or service	<code>install-firmware</code>	Yes	Changes the update channel and the automatic installation of the controller or of a device, or postpones an update (Security and Access Sections 15.5 and 15.7).
<code>updates.revert</code>	request	app or service	<code>install-firmware</code>	Yes	Returns the controller or a device to its previous firmware (Security and Access Section 15.4.4).
<code>access.readRemote</code>	request	app or service	<code>view-configuration</code>	—	Returns whether remote access is on, and its relays (Security and Access Section 13).
<code>access.configureRemote</code>	request	app or service	<code>configure-remote-access</code>	Yes	Turns remote access on or off, and sets its relays.
<code>access.readEmergency</code>	request	app or service	<code>view-configuration</code>	—	Returns whether emergency access is on and available, how to join its link, and its emergency codes (Security and Access Section 14).
<code>access.configureEmergency</code>	request	app or service	<code>configure-emergency-access</code>	Yes	Turns emergency access on or off, or allows emergency codes again after failed attempts.
<code>access.createEmergencyCode</code>	request	app or service	<code>configure-emergency-access</code>	Yes	Creates an emergency code for a person without an app,

Method	Type	From	Operation	Audited	Description
					such as a neighbor, as a new user with the <code>override</code> role (Security and Access Section 14.4.2).
<code>access.removeEmergencyCode</code>	request	app or service	<code>configure-emergency-access</code>	Yes	Removes an emergency code, and its user.
<code>privacy.listConsent</code>	request	app or service	<code>view-configuration</code>	—	Returns each purpose for which the controller, or a device, would send data off the owner's network, and whether an owner has consented (Security and Access Section 16.3).
<code>privacy.setConsent</code>	request	app or service	<code>manage-consent</code>	Yes	Gives or withdraws an owner's consent for one purpose, with effect at once.
<code>privacy.eraseData</code>	request	app or service	<code>erase-data</code>	Yes	Erases the controller's history and other personal data, without a factory reset (Security and Access Section 16.6.2).

7.10 Extensions

1. A controller MAY serve extension methods, named as Device Protocol Section 9.9.1 says, such as `com.example.reef.photoMode`. Extensions to the members of a method's parameters or result go in their `extensions` [OAP-DP Section 9.3.5].
2. Each extension method that a controller serves MUST name an operation of Security and Access Table 6, MUST be checked against it as Section 7.2 requires, and, when that operation is audited, MUST be recorded as Section 7.3 requires. An extension method is never a way around a role.

7.11 Requests That Reach Devices

Some requests are carried out by commands or requests to a device, such as an action (Section 15), a calibration (Section 16), or a change of settings (Section 8.6). The controller sends them itself, as the device's controller, with the messages of the device's binding [OAP-DP Section 11; OAP-MB Section 11], and the app never reaches the device.

1. A controller MUST respond to such a request only once the device has accepted or refused what the controller sent it, and MUST refuse the request with `bad-state` and the reason `device-refused`, with the device's error as `error`, when the device refused it.
2. While it has lost the device, a controller MUST refuse such a request with `bad-state` and the reason `device-lost`, and MUST NOT keep it to send later [OAP-DP Section 12.4.4].
3. When the device has not answered within 4 s (provisional), a controller MUST respond with `bad-state` and the reason `device-unanswered`, so that it responds within the time that the Device Protocol allows [OAP-DP Section 9.5.2], and MUST NOT send the command again until it has read the device's state and events [OAP-DP Section 12.4.3].
4. A controller MUST NOT send such a command, other than one that stops an output, to an endpoint under local control, and refuses the request with `local-control` and the reason `under-local-control` [OAP-AU Section 17.4.2].
5. The stops that a controller sends for its control functions, interlocks, and `stop-all` go ahead of what it sends for apps [OAP-DP Section 9.5.4].

7.12 Long Operations

1. `system.addDevice` and `updates.install` start operations that may take longer than a response may wait [OAP-DP Section 9.5.2]. A controller responds to such a request at once, with the operation's identifier, and then carries out the operation.

2. A controller MUST report the end of each operation on the session that started it, while that session lasts, with `system.deviceAdded` or with the last `updates.progress`. An operation goes on when the session ends, and an app learns its outcome from `system.listDevices` or `updates.list`.
3. The error of an operation that fails has a code of the Device Protocol, and a reason of Table 2, as Section 7.4 says.

8 The System

8.1 The Controller

system.describe: request, from an app or service. Returns the controller's identity, the revisions of the specifications it implements, and its clock. Operation: `view-configuration`.

Parameters: none.

Result	Type	Presence	Description
<code>keyId</code>	string	REQUIRED	The controller's key ID (Security and Access Section 8.3).
<code>name</code>	string	OPTIONAL	The controller's name for people, as its alarm messages give it.
<code>manufacturer</code>	string	REQUIRED	The controller's maker.
<code>model</code>	string	REQUIRED	The controller's model.
<code>firmwareVersion</code>	string	REQUIRED	The version of the firmware it runs.
<code>serialNumber</code>	string	OPTIONAL	Its serial number.
<code>specRevisions</code>	object	REQUIRED	The revision of each Open Aquarium specification it implements, by spec-id, such as <code>{"controller-api": "0.1", "automation": "0.3"}</code> . It includes <code>controller-api</code> .
<code>time</code>	timestamp or null	REQUIRED	Its time now, or null while its clock is not valid (Fail-Safe Behavior Section 10.5).
<code>timeZone</code>	string	OPTIONAL	The time zone of its automation configuration (Automation Section 7.2).
<code>uptime</code>	number (s)	REQUIRED	The time since it last started.

1. `specRevisions` MUST include each Open Aquarium specification that the controller implements, with the revision that it implements, and `controller-api` with `0.1` for this revision.
2. An app finds what a controller supports from `specRevisions`, rather than from its model. A controller responds to a method that it does not serve with `method-not-found` [OAP-DP Section 9.5.7], so that an app can work with a controller of an earlier revision.

For example:

```
{ "id": 3, "method": "system.describe" }
{ "id": 3, "result": { "keyId": "q8mB0x4yZp2Vt7LrKc3sHw9dNf1gJa6eUo5iXb2lTvE", "name": "Reef controller",
  "manufacturer": "Example Aquatics", "model": "C1", "firmwareVersion": "1.4.0",
  "specRevisions": { "controller-api": "0.1", "device-protocol": "0.3", "security-and-access": "0.3",
    "automation": "0.3", "alarms-and-notifications": "0.3", "data-model": "0.4" },
  "time": "2026-10-18T18:00:00+10:00", "timeZone": "Australia/Brisbane", "uptime": 86400 } }
```

8.2 Devices and Modules

system.listDevices: request, from an app or service. Returns the controller's devices and modules, with their endpoints. Operation: `view-configuration`.

Parameters: none.

Result	Type	Presence	Description
<code>devices</code>	array	REQUIRED	Each device and module: its device, as Automation Section 8.5 identifies it; its manufacturer, model, name, and firmwareVersion; lost, which is true while the controller has lost it (Fail-Safe Behavior Section 7.6); and its endpoints, each with its id, class, features, and name.

system.describeDevice: request, from an app or service. Returns the description of one device or module. Operation: `view-configuration`. Reasons it may give (Table 2): `bad-endpoint` with `unknown-device`.

Parameter	Type	Presence	Description
device	string	REQUIRED	The device, as Automation Section 8.5 identifies it.

Result	Type	Presence	Description
description	object	REQUIRED	Its device description as the controller last read it, valid against the Device Classes schema (Device Classes Section 12).

1. `system.listDevices` MUST return every device and module that the controller has, whether or not it has lost them: those that its configuration binds, those that it has paired or accepted, and its own built-in endpoints, as `controller:self` [OAP-AU Section 8.5].
2. `lost` is `true` from when the controller treats a device as lost [OAP-FS Section 7.6], and a bridged device as unreachable [OAP-DP Section 12.5], until it hears from the device again.
3. `system.describeDevice` MUST return the description that the controller last read from the device, valid against the Device Classes schema [OAP-DC Section 12], and, for `controller:self`, a description of the controller's own endpoints.

8.3 State

system.read: request, from an app or service. Returns the current values of attributes of a device or of one of its endpoints, as the controller last received them. Operation: `view-state`. Reasons it may give (Table 2): `bad-endpoint` with `unknown-device`; `bad-attribute` with `unknown-attribute`.

Parameter	Type	Presence	Description
device	string	REQUIRED	The device, as Automation Section 8.5 identifies it.
endpoint	string	OPTIONAL	The endpoint. Without it, the device attributes (Device Classes Section 8.1).
attributes	array	OPTIONAL	The names of the attributes to read. Without it, every config and state attribute.

Result	Type	Presence	Description
attributes	object	REQUIRED	The value of each attribute, by name, encoded as Device Protocol Section 9.7 says.
time	timestamp or <code>null</code>	REQUIRED	When the device reported them, or <code>null</code> when it did not give a valid time.
current	boolean	REQUIRED	<code>false</code> while the controller has lost the device, so that the values are the last it reported.

system.subscribe: request, from an app or service. Starts `system.state` and `system.devicesChanged` notifications on the session, replacing any earlier subscription of the `system` namespace. Operation: `view-state`. Reasons it may give (Table 2): `bad-endpoint` with `unknown-device`.

Parameter	Type	Presence	Description
devices	array	OPTIONAL	The devices whose state to report. Without it, every device.

Result: `{}`.

system.unsubscribe: request, from an app or service. Ends the session's subscription of the `system` namespace. Operation: `view-state`.

Parameters: none.

Result: `{}`.

system.state: notification, from the controller. Reports the state of a device or of one of its endpoints.

Parameter	Type	Presence	Description
device	string	REQUIRED	The device.
endpoint	string	OPTIONAL	The endpoint. Absent in the report of the device's own attributes.
time	timestamp or <code>null</code>	REQUIRED	When the device reported the state, or <code>null</code> when it did not give a valid time.
attributes	object	REQUIRED	The current value of every state attribute, including <code>faults</code> , and <code>controlSource</code> for output and unit endpoints.

Parameter	Type	Presence	Description
current	boolean	REQUIRED	false while the controller has lost the device.

system.devicesChanged: notification, from the controller. Reports that the controller's devices, or whether it has lost one, changed.

Parameter	Type	Presence	Description
devices	array	REQUIRED	Every device and module, as <code>system.listDevices</code> returns them.

1. `system.read` and `system.state` give the values that the controller last received from the device, and `system.read` refuses an attribute that the endpoint or device does not have with `bad-attribute` and the reason `unknown-attribute`.
2. A controller MUST NOT send a device any message in order to carry out `system.read` or a subscription of the `system` namespace, so that apps never add to the traffic in front of a stop [OAP-DP Section 9.5.4].
3. After it responds to `system.subscribe`, a controller MUST send `system.state` for each device of the subscription and for each of its endpoints, and then MUST send `system.state` for a device or an endpoint within 1 s of receiving a change of its state, but not more than once a second for each, merging the changes in between.
4. A `system.state` MUST carry every state attribute of the device or endpoint, so that a report that replaces a missed or merged one loses nothing [OAP-DP Section 11.5.4].
5. A controller MUST send `system.devicesChanged` within 1 s when it gains or removes a device, or when a device's `lost` changes.

For example, the state of a heater outlet:

```
{ "method": "system.state",
  "params": { "device": "device-protocol:6dNMJo2ewLHXnPgVjZ31VrRskHpj6lqSDjEMsUK92h8", "endpoint": "outlet_1",
    "time": "2026-10-18T18:00:03.250+10:00", "current": true,
    "attributes": { "controlSource": "controller", "latched": false, "faults": [], "on": true } } }
```

8.4 Identifying

system.identify: request, from an app or service. Makes a device, or one of its endpoints, show where it is, without changing any output. Operation: `identify`. Reasons it may give (Table 2): `bad-endpoint` with `unknown-device`; `bad-state` with `device-lost` and `device-refused`.

Parameter	Type	Presence	Description
device	string	REQUIRED	The device, as Automation Section 8.5 identifies it.
endpoint	string	OPTIONAL	The endpoint. Without it, the whole device.
duration	number (s)	OPTIONAL	How long it shows where it is, from 1 s to 300 s. Default 10 s.

Result: `{}`.

1. `system.identify` makes the endpoint show where it is with its `identify` command, or, without `endpoint`, the whole device with `device.identify` [OAP-DP Section 11.4.3], or, on the module bus, with `identify` on each of its endpoints, for `duration`.
2. A controller MUST NOT change any output to carry it out.

A Matter bridge on another host carries out a home app's Identify with this method [OAP-MM Table 8].

8.5 Adding and Removing Devices

system.discover: request, from an app or service. Returns the units that the controller finds with a pairing window open, which an owner can add. Operation: `view-configuration`.

Parameters: none.

Result	Type	Presence	Description
units	array	REQUIRED	Each unit that advertises a pairing window on the controller's networks (Device Protocol Section 8.3.2): its <code>keyId</code> , its instance <code>name</code> , <code>kind</code> (device or controller), its <code>discriminator</code> , where it advertises one, and its addresses.

system.addDevice: request, from an app or service. Starts adding a network device to the system: claiming it with its setup code, pairing with an invitation it made, or taking on a device that an owner's app has claimed. Operation: `add-device`, and also `change-network-settings` with `deviceNetwork` `true`. Audit record: `add-device`, with the device as `target.device` once it is known, and how it was added, `setup-code`, `invitation`, or `key`, in `details`, as `method`; never the code. Reasons it may give (Table 2): `bad-state` with `already-added` and `not-found`.

Parameter	Type	Presence	Description
<code>setupCode</code>	string	One of	The device's setup code: the text of its QR code, such as <code>0AP:1:2817:30184762</code> , or its digits.
<code>invitation</code>	string	One of	An invitation that an owner's app created on the device, such as <code>27-4815-1623</code> (Device Protocol Section 10.5).
<code>keyId</code>	string	One of, or with a code	The key ID of the device: alone, for a device whose owner's app has added the controller to its pairing list (Security and Access Section 9.5.5); with a code, to choose among the units that <code>system.discover</code> returns.
<code>address</code>	string	OPTIONAL	The device's address, entered by hand, for a device that discovery does not reach (Device Protocol Section 8.5.3).
<code>port</code>	integer	With <code>address</code>	The port of its sessions.
<code>deviceNetwork</code>	boolean	OPTIONAL	<code>true</code> to move the device onto the controller's own Wi-Fi network once it is claimed (Device Protocol Section 14.1.3), which also needs <code>change-network-settings</code> . Default <code>false</code> .

Result	Type	Presence	Description
<code>operation</code>	string	REQUIRED	The operation's identifier, which <code>system.deviceAdded</code> reports (Section 7.12).

system.deviceAdded: notification, from the controller. Reports the end of an operation of `system.addDevice`, to the session that started it.

Parameter	Type	Presence	Description
<code>operation</code>	string	REQUIRED	The operation.
<code>device</code>	string	On success	The device added.
<code>replaces</code>	string	OPTIONAL	A device that the controller has lost, with the same manufacturer, model, and serial number, whose bindings the owner may move to the new one (Device Protocol Section 7.3.4).
<code>error</code>	object	On failure	Why the device was not added, as an error of the Device Protocol, with <code>data</code> as Section 7.4 says.

system.removeDevice: request, from an app or service. Removes a device or module from the system: the controller releases it, factory resets it if asked, and leaves its pairing list. Operation: `remove-device`, and also `factory-reset` with `factoryReset` `true`. Audit record: `remove-device`, with the device as `target.device`, and `factoryReset` in `details` when it is `true`; a factory reset is also recorded as `factory-reset`. Reasons it may give (Table 2): `bad-endpoint` with `unknown-device`; `bad-state` with `device-refused` and `device-unanswered`.

Parameter	Type	Presence	Description
<code>device</code>	string	REQUIRED	The device.
<code>factoryReset</code>	boolean	OPTIONAL	<code>true</code> to factory reset the device before leaving it, which also needs <code>factory-reset</code> (Security and Access Section 9.9.2). Default <code>false</code> .

Result: `{}`.

1. `system.discover` returns the units that advertise a pairing window on the controller's networks, as the controller finds them by browsing for the `_pairing` subtype [OAP-DP Section 8.3.2], so that the app can show the person what may be added.
2. `system.addDevice` claims a network device as Security and Access Section 9.5 says, which a controller MUST do only for a user whose role permits `add-device` [OAP-SA Section 9.5.3]:
 - with `setupCode`, it pairs with the unit of `keyId`, or with a unit whose pairing window is open and whose discriminator is the code's, and MUST NOT try the code on a unit whose discriminator differs [OAP-DP Section 10.4.5];

- with `invitation`, it pairs with the invitation that an owner's app created on the device [OAP-DP Section 10.5];
 - with `keyId` alone, it takes on the device whose owner's app has added the controller to its pairing list [OAP-SA Section 9.5.5], which proves its key in the handshake of its session [OAP-DP Section 10.1]; and
 - with `address` and `port`, it connects to them, rather than finding the device by discovery [OAP-DP Section 8.5.3].
3. Once it is paired, a controller MUST add the owners' keys to the device's pairing list as Security and Access Section 9.5.4 requires, take control of the device [OAP-DP Section 12.1.2], read its description, subscribe to it, and resume control of each endpoint as the Device Protocol says [OAP-DP Section 12.4], and, with `deviceNetwork true`, move it onto its own Wi-Fi network [OAP-DP Section 14.1.3].
 4. The controller reports the end of the operation with `system.deviceAdded` (Section 7.12): the device, on success, and, for a device with the same manufacturer, model, and serial number as one that it has lost, `replaces`, so that the app can offer to move the lost device's bindings, after the owner confirms [OAP-DP Section 7.3.4; OAP-DC Section 7.12]; or the error, such as `pairing-failed` [OAP-DP Section 10.2.4], or `bad-state` with the reason `not-found`.
 5. A controller MUST NOT put a setup code, or the code of an invitation, in a result, a notification, or an audit record [OAP-SA Section 9.2.4].
 6. `system.removeDevice` releases the device [OAP-DP Section 12.1.4; OAP-MB Section 11.7], factory resets it, when `factoryReset` is `true`, and then leaves its pairing list [OAP-DP Section 10.6]. A controller MUST do these in that order, so that the device's endpoints take their fail-safe behavior at once, rather than at the end of their `failSafeTimeout` [OAP-DP Section 10.6.4], and MUST then forget the device, whose bindings become unusable [OAP-AU Section 8.5.7]. A device that it has lost it forgets without sending it anything.
 7. An app that removes a device MUST offer to factory reset it first, as Security and Access Section 9.9.2 requires of a controller, and SHOULD offer to move its bindings to another device [OAP-DC Section 7.12.1].

Modules join the module bus, and are configured, by themselves [OAP-MB Section 10.1], and `system.listDevices` lists them. A module that `system.removeDevice` removed, but that is still on the bus, joins again, so an owner takes a module off the bus before removing it.

8.6 Device Settings

`system.writeSettings`: request, from an app or service. Changes config attributes of a device or of one of its endpoints, all or none. Operation: `change-device-settings`, or `change-fail-safe-settings` for fail-safe settings, and `install-firmware` for `updateChannel` and `automaticInstall`, each as the attributes need (Section 8.6). Audit record: `change-device-settings`, or `change-fail-safe-settings` or `install-firmware`, one record for each operation it needed, with the device and the endpoint as `target.device` and `target.endpoint`, and each change in `details`, as `changes`, each with its `attribute` and its values `from` and `to`; a check is not recorded. Reasons it may give (Table 2): `bad-endpoint` with `unknown-device`; `bad-attribute` with `unknown-attribute`; `bad-parameter` with `confirmation-needed` and `protected-setting`; `bad-state` with `device-lost`, `device-refused`, and `device-unanswered`.

Parameter	Type	Presence	Description
<code>device</code>	string	REQUIRED	The device.
<code>endpoint</code>	string	OPTIONAL	The endpoint. Without it, the device attributes.
<code>attributes</code>	object	REQUIRED	The new value of each attribute to change, by name, encoded as Device Protocol Section 9.7 says.
<code>confirmed</code>	boolean	OPTIONAL	<code>true</code> when the person has confirmed a change of fail-safe settings, as Fail-Safe Behavior Section 13.1.3 requires (Section 8.6). Default <code>false</code> .
<code>check</code>	boolean	OPTIONAL	<code>true</code> to check the change without making it. Default <code>false</code> .

Result	Type	Presence	Description
<code>attributes</code>	object	REQUIRED	The value of each attribute given, as it is after the request, or would be.
<code>recommended</code>	object	OPTIONAL	For a change of <code>loadCategory</code> , the recommended fail-safe configuration of the new category, as fail-safe attributes, for the person to confirm (Fail-Safe Behavior Section 13.1.2).

Result	Type	Presence	Description
warnings	array	REQUIRED	What the controller warns the owner of (Section 8.6).

1. `system.writeSettings` writes config attributes of a device or of one of its endpoints, as the device's controller. It needs `change-fail-safe-settings` for `failSafeAction`, `failSafeState`, `failSafeTimeout`, `powerOnBehavior`, `powerOnDelay`, `loadCategory`, and `backupThermostat`; `install-firmware` for `updateChannel` and `automaticInstall`; and `change-device-settings` for every other config attribute, names and safety limits among them [OAP-DP Section 11.2.7; OAP-SA Table 6]. A write of attributes of several of these kinds needs each of their operations.
2. A controller MUST carry out a write completely or not at all [OAP-DP Section 11.2.4]. On a module, whose registers it writes one at a time, it checks every value against the module's limits first, writes them in an order that the module accepts, and writes back the previous values when a write fails [OAP-MB Section 11.2].
3. A controller MUST refuse a change of fail-safe settings without `confirmed true`, with `bad-parameter` and the reason `confirmation-needed`, because the owner's confirmation, at the time of the change, is required [OAP-FS Section 13.1.3; OAP-SA Section 11.3.4; OAP-PM Section 13.3.1]. An app sends `confirmed true` only once the person has confirmed the change, with the warnings of a check.
4. A controller MUST refuse, with `bad-parameter` and the reason `protected-setting`, a change that the specifications forbid it to make, such as `failSafeAction continue` for a return pump [OAP-FP Section 16.1.1], and a change of an attribute that it writes itself from the configuration, such as a light's or a pump's program.
5. A check and a write MUST return as warnings what the specifications require a controller to warn the owner of about the change, among them a fail-safe configuration that differs from the recommendation for the output's `loadCategory` [OAP-FS Section 13.3.2], a safety function left without a path that uses no radio link [OAP-FS Section 13.4.6], and `continue` for a top-off unit whose water carries an additive [OAP-WL Section 8.16.7], and, for a change of `loadCategory`, MUST return `recommended`, the recommended configuration of the new category [OAP-FS Section 13.1.2].
6. When a change disables the backup thermostat of an outlet whose `failSafeAction` is `continue`, or removes its `setpoint`, a controller MUST first set its `failSafeAction` to `state`, with `failSafeState {"on": false}`, as part of the same confirmed change [OAP-PM Section 13.3.4].
7. A controller records a change of fail-safe settings with its old and new values [OAP-SA Section 12.1.2], as the audit record of `system.writeSettings` says.

8.7 Restarting and Factory Reset

system.restart: request, from an app or service. Restarts a device, or the controller itself. Operation: `restart`. Audit record: `restart`, with the device as `target.device`, or no target for the controller. Reasons it may give (Table 2): `bad-endpoint` with `unknown-device`; `bad-state` with `operation-running`, `device-lost`, `device-refused`, and `device-unanswered`.

Parameter	Type	Presence	Description
device	string	OPTIONAL	The device. Without it, the controller.

Result: `{}`.

system.factoryReset: request, from an app or service. Factory resets the controller (Security and Access Section 9.8). Operation: `factory-reset`. Audit record: `factory-reset`, which the factory reset then erases with the rest of the audit log (Security and Access Section 12.3.2).

Parameters: none.

Result: `{}`.

1. With `device`, `system.restart` restarts the device [OAP-DP Section 13.4.1], and a controller MUST refuse it, with `bad-state` and the reason `operation-running`, while one of the device's endpoints is dosing, filling, or taking part in a water change, with the meanings of Fail-Safe Behavior Section 11.2.2.

- Without `device`, the controller restarts itself once it has responded, closing its sessions with code 4005 [OAP-DP Table 4], and starts again as Fail-Safe Behavior Section 9.4 and Automation Section 17.7 say. A restart that a user asked for raises no notice [OAP-AN Section 13.3.3].
- `system.factoryReset` factory resets the controller once it has responded, as Security and Access Section 9.8 says, closing every session with code 4003: it erases its users, credentials, configuration, history, audit log, and every other piece of owner data, and its devices, which lose their controller, take their fail-safe behavior [OAP-SA Section 9.8.4]. A device is factory reset as it is removed, with `system.removeDevice`.
- An app SHOULD ask the person to confirm again before `system.factoryReset` [OAP-SA Section 11.3.6].

9 History

9.1 Reading History

`history.read`: request, from an app or service. Returns the history of a time range as a Data Model Document, one page at a time. Operation: `view-history`. Reasons it may give (Table 2): `bad-parameter` with `unknown-element` and `unknown-cursor`.

Parameter	Type	Presence	Description
<code>from</code>	timestamp	REQUIRED	The start of the range.
<code>to</code>	timestamp	OPTIONAL	The end of the range, not included. Default now.
<code>systems</code>	array	OPTIONAL	Only these systems. Default every system.
<code>include</code>	array	OPTIONAL	Only these members of a Document: <code>observations</code> , <code>series</code> , <code>samples</code> , <code>events</code> , <code>states</code> , <code>ranges</code> , <code>tasks</code> , and <code>livestock</code> . Default all.
<code>parameters</code>	array	OPTIONAL	Only observations and series of these Data Model parameters.
<code>eventTypes</code>	array	OPTIONAL	Only events of these types.
<code>cursor</code>	string	OPTIONAL	The <code>next</code> of the previous page, with the other parameters unchanged.

Result	Type	Presence	Description
<code>document</code>	object	REQUIRED	A Data Model Document of revision 0.4 (Section 9).
<code>next</code>	string	OPTIONAL	Present when more of the range remains: the <code>cursor</code> of the next page.

- `history.read` MUST return a Data Model Document of revision 0.4 that meets every requirement of the Data Model for Documents [OAP-DM Section 10.1], so that every system, vessel, equipment, and sample that its records refer to is in the Document [OAP-DM Section 7.3].
- It holds the records of the systems asked for whose times lie in the range: observations and samples by their time, the values of series and state series that lie in the range, and events by their `time`.
- The owner's ranges, tasks, and livestock, which describe the systems rather than what happened in the range, are in the first page, whatever the range.
- Closed alarms are events of the type `alarm` [OAP-AN Section 20.1], so `eventTypes ["alarm"]` reads the history of alarms; alarms that are not closed come from `alarms.list` (Section 11.1).

9.2 Pages

- A controller MAY split a range into pages, each a whole Document. When more of the range remains after a page, its result has `next`, and the app asks for the next page with `next` as `cursor`, and the other parameters unchanged.
- A controller MUST return the records of a range in pages in order of their times, MUST NOT return a record in two pages of one range, and MUST accept a `next` that it gave for at least 10 minutes (provisional) after it gave it.
- A controller refuses a `cursor` that it did not give, or that it no longer accepts, with `bad-parameter` and the reason `unknown-cursor`, and the app then reads the range again from where it got to.

For example, the first page of a day of temperatures, with its values shortened:

```
{ "id": 40, "method": "history.read",
  "params": { "from": "2026-10-17T00:00:00+10:00", "to": "2026-10-18T00:00:00+10:00",
    "include": ["series"], "parameters": ["temperature"] } }
{ "id": 40, "result": {
  "document": { "oapDataModel": "0.4", "created": "2026-10-18T18:00:00+10:00",
    "systems": [{ "id": "reef", "name": "Living room reef", "water": "marine",
      "vessels": [{ "id": "sump", "role": "sump" }] }],
    "series": [{ "parameter": "temperature", "unit": "degC", "system": "reef", "vessel": "sump",
      "start": "2026-10-17T00:00:00+10:00", "interval": 300,
      "values": [25.6, 25.6, 25.5, 25.5, 25.4] } ] },
  "next": "c2VyaWVzOjIwMjYtMTA6MTdUMTI6MDA6MDArMTA6MDA" } }
```

10 Automation

10.1 The Configuration Document

automation.readConfiguration: request, from an app or service. Returns the automation configuration document. Operation: `view-configuration`.

Parameter	Type	Presence	Description
<code>state</code>	boolean	OPTIONAL	<code>true</code> to include the saved state, as <code>state</code> . Default <code>false</code> .

Result	Type	Presence	Description
<code>document</code>	object	REQUIRED	The configuration document (Automation Section 7).
<code>version</code>	string	REQUIRED	The document's version. A string of 1 to 64 characters that changes whenever the document changes (Section 7.6).

automation.validateConfiguration: request, from an app or service. Checks a configuration document as `automation.replaceConfiguration` would, without changing anything. Operation: `view-configuration`.

Parameter	Type	Presence	Description
<code>document</code>	object	REQUIRED	The configuration document to check.

Result	Type	Presence	Description
<code>valid</code>	boolean	REQUIRED	Whether the controller would accept the document.
<code>errors</code>	array	REQUIRED	Why it would not, each a <code>message</code> for people and, where it applies, the <code>path</code> of the member, as a JSON Pointer.
<code>warnings</code>	array	REQUIRED	What the controller would warn the owner of if it ran the document (Section 7.6).

automation.replaceConfiguration: request, from an app or service. Replaces the automation configuration document. Operation: `change-configuration`. Audit record: `change-configuration`, with the previous and the new `version` in details, as `from` and `to`. Reasons it may give (Table 2): `bad-parameter` with `invalid-document`; `bad-state` with `version-conflict`; `not-permitted` with `not-bridge`.

Parameter	Type	Presence	Description
<code>document</code>	object	REQUIRED	The new configuration document.
<code>ifVersion</code>	string	OPTIONAL	Replace it only if the current document has this version.
<code>fabric</code>	object	OPTIONAL	From a Matter bridge's user only: the Matter fabric for which the bridge asks, as <code>fabricIndex</code> , <code>label</code> , and <code>vendorId</code> (Section 10.9).

Result	Type	Presence	Description
<code>version</code>	string	REQUIRED	The new document's version.
<code>warnings</code>	array	REQUIRED	What the controller warns the owner of, as <code>automation.validateConfiguration</code> returns it.

1. `automation.readConfiguration` MUST return the complete configuration document, as Automation Section 19.1 requires of an export, with every element, binding, and extension that the controller holds, and with the saved state only when `state` is `true`.

2. `automation.validateConfiguration` MUST check a document as Automation Section 19.2 requires a controller to check one that it imports, and as the specifications of its hosted functions require [OAP-AU Section 11.10], and MUST return each element that the controller does not support as a warning `unsupported` [OAP-AU Section 19.2.3], and each output and endpoint input without a usable binding as a warning `binding-missing` [OAP-AU Section 19.2.4].
3. A controller MUST refuse, with `bad-parameter` and the reason `invalid-document`, an `automation.replaceConfiguration` whose document it would not accept, with the problems in `errors`, and MUST then keep its running configuration [OAP-AU Section 19.2.1].
4. When it accepts a document with `state`, a controller MUST take that saved state in place of its own, as an import does, so that modes, overrides, and programs continue [OAP-AU Section 19.2.6]. When it accepts a document without `state`, it MUST keep each part of its saved state whose element the new document still has, and MUST end each other part as a person would end it, so that, for example, removing an active mode ends the mode.
5. Once it has accepted a document, a controller runs it at its next evaluation [OAP-AU Section 17.1], gives it a new version, and reports it on every session subscribed to the `automation` namespace (Section 10.2).

An app that edits the configuration reads it with its version, changes it, checks it, shows the person the warnings, and replaces it with `ifVersion` (Section 7.6). A Matter bridge on another host changes setpoint offsets and names in the same way (Section 10.9).

10.2 Saved State and Status

`automation.readState`: request, from an app or service. Returns the saved state, and what decides each output now. Operation: `view-state`.

Parameters: none.

Result	Type	Presence	Description
<code>version</code>	string	REQUIRED	The configuration document's version.
<code>time</code>	timestamp or <code>null</code>	REQUIRED	The controller's time, or <code>null</code> while its clock is not valid.
<code>state</code>	object	REQUIRED	The saved state (Automation Section 17.8).
<code>outputs</code>	array	REQUIRED	Each output's status (Table 4).

`automation.subscribe`: request, from an app or service. Starts `automation.changed` notifications on the session. Operation: `view-state`.

Parameters: none.

Result: `{}`.

`automation.unsubscribe`: request, from an app or service. Ends the session's subscription of the `automation` namespace. Operation: `view-state`.

Parameters: none.

Result: `{}`.

`automation.changed`: notification, from the controller. Reports that the saved state, the status of an output, or the configuration document changed.

Parameter	Type	Presence	Description
<code>version</code>	string	REQUIRED	The configuration document's version.
<code>time</code>	timestamp or <code>null</code>	REQUIRED	The controller's time, or <code>null</code> while its clock is not valid.
<code>state</code>	object	REQUIRED	The saved state.
<code>outputs</code>	array	REQUIRED	Each output's status.

1. `automation.readState` MUST return the saved state as Automation Section 17.8 defines it, and the status of every output of the configuration, virtual outputs included, with the members of Table 4.
2. An output's status MUST give what Automation Section 17.10 requires a controller to show: its resolved state, the layer that decided it, as one of Table 5, the element of that layer, and the time left of a temporary state.

- After it responds to `automation.subscribe`, a controller MUST send `automation.changed` within 1 s of each change of the saved state, of an output's status other than its `left`, or of the configuration document's version, but not more than once a second, merging the changes in between.

Table 4 — Members of an output's status

Member	Type	Presence	Description
<code>output</code>	identifier	REQUIRED	The output.
<code>state</code>	state, "fail-safe", or <code>null</code>	REQUIRED	Its resolved state: run, stop, or a level (Automation Table 4); "fail-safe" for a function's unavailable-input behavior that asks for the output's fail-safe behavior [OAP-AU Section 11.9]; or <code>null</code> when no layer has a demand [OAP-AU Section 17.3.3].
<code>layer</code>	string	REQUIRED	The layer of Automation Table 25 that decided it, as Table 5 names it.
<code>element</code>	identifier	OPTIONAL	The element of that layer that decided it: the interlock, the mode, the step program, or the function. Absent for <code>stop-all</code> , an override, and <code>none</code> .
<code>left</code>	number (s)	OPTIONAL	How long until the state ends: an override, <code>stop-all</code> , a mode with a duration or its extended state, or a program's step.
<code>until</code>	timestamp	OPTIONAL	When it ends, while the controller's clock is valid.
<code>latched</code>	boolean	OPTIONAL	<code>true</code> when the controller latched the stop [OAP-AU Section 17.5.4].
<code>controlSource</code>	string or <code>null</code>	OPTIONAL	What the endpoint reports it follows: <code>controller</code> , <code>local</code> , or <code>fail-safe</code> [OAP-DC Section 8.3]; <code>null</code> while the controller does not know, such as while it has lost the device.
<code>extensions</code>	object	OPTIONAL	Extensions.

Table 5 — Layers

layer	Layer of Automation Table 25	element
<code>interlock</code>	1, an interlock, or a check of a hosted function at the precedence of interlocks [OAP-AU Section 11.10.6]	The interlock or the function
<code>stop-all</code>	1, <code>stop-all</code>	—
<code>emergency</code>	2	The emergency mode
<code>power-loss</code>	3	The power-loss mode
<code>override</code>	4	—
<code>water-change</code>	5	The water-change mode
<code>maintenance</code>	6	The maintenance mode
<code>feed</code>	7	The feed mode
<code>mode</code>	8, a mode of an extension kind	The mode
<code>program</code>	9	The step program
<code>function</code>	10	The control function, including a hosted function
<code>none</code>	11	—

When an interlock and `stop-all` both stop an output, its `layer` is `interlock`. A mode's extended state has the mode's layer and element [OAP-AU Section 14.4.3]. The overrides, active modes, programs, and `stop-all` are in the saved state, each with the user who started it, which a Matter bridge needs to tell a person's override or mode from its own [OAP-MM Section 12.4].

10.3 Modes

`automation.startMode`: request, from an app or service. Starts a mode, or starts an active mode again.

Operation: `start-mode`. Audit record: `start-mode`, with the mode as `target.element`, and `restart` in `details` when it is `false`. Reasons it may give (Table 2): `bad-parameter` with `unknown-element`; `bad-state` with `not-by-person` and `unsupported`; `not-permitted` with `not-bridge`.

Parameter	Type	Presence	Description
<code>mode</code>	identifier	REQUIRED	The mode.
<code>restart</code>	boolean	OPTIONAL	<code>false</code> to change nothing if the mode is active already, rather than move its end (Section 10.9). Default <code>true</code> .

Parameter	Type	Presence	Description
fabric	object	OPTIONAL	From a Matter bridge's user only: the Matter fabric for which the bridge asks, as <code>fabricIndex</code> , <code>label</code> , and <code>vendorId</code> (Section 10.9).

Result	Type	Presence	Description
left	number (s)	OPTIONAL	How long until it ends. Absent when it has no end.
until	timestamp	OPTIONAL	When it ends. Absent when it has no end, and while the controller's clock is not valid.

automation.endMode: request, from an app or service. Ends a mode. Operation: `start-mode`, or `end-emergency-mode` to end an emergency mode. Audit record: `start-mode`, or `end-emergency-mode` for an emergency mode, with the mode as `target.element`, and `{"end": true}` in `details`. Reasons it may give (Table 2): `bad-parameter` with `unknown-element`; `bad-state` with `not-by-person` and `other-user`; `not-permitted` with `not-bridge`.

Parameter	Type	Presence	Description
mode	identifier	REQUIRED	The mode.
onlyOwn	boolean	OPTIONAL	<code>true</code> to carry out the request only if no other user set the override or started the mode (Section 10.9). Default <code>false</code> .
fabric	object	OPTIONAL	From a Matter bridge's user only: the Matter fabric for which the bridge asks, as <code>fabricIndex</code> , <code>label</code> , and <code>vendorId</code> (Section 10.9).

Result: `{}`.

1. `automation.startMode` starts a mode as an owner starts it [OAP-AU Section 14.3]: it ends any active mode of the same kind that may not be active with it, and a mode that is active already starts again, with its end moved to its `duration` from now, unless `restart` is `false`.
2. A controller MUST record the session's user as the user who started the mode [OAP-AU Section 14.3.7].
3. A controller MUST refuse to start or end a `power-loss` mode, with `bad-state` and the reason `not-by-person` [OAP-AU Section 14.6.3], and MUST refuse to start a mode that it does not support, with the reason `unsupported` [OAP-AU Section 19.2.3].
4. `automation.endMode` needs `end-emergency-mode` for an emergency mode, and `start-mode` for any other [OAP-SA Table 6]. Ending a mode starts its extended states [OAP-AU Section 14.4.3].
5. Ending a mode that is not active changes nothing, and succeeds.

Starting a water-change mode starts the water change that names it [OAP-WL Section 9.4].

10.4 Scenes

automation.startScene: request, from an app or service. Starts a scene of a lighting function. Operation: `start-mode`. Audit record: `start-mode`, with the lighting function as `target.element`, and `scene` and `duration` in `details`. Reasons it may give (Table 2): `bad-parameter` with `unknown-element` and `duration-too-long`; `not-permitted` with `not-bridge`.

Parameter	Type	Presence	Description
function	identifier	REQUIRED	The lighting function.
scene	identifier	REQUIRED	The scene, of the function's program.
duration	integer (s)	OPTIONAL	How long the scene lasts, if less than its own <code>duration</code> (Lighting Programs Section 13.2.1).
fabric	object	OPTIONAL	From a Matter bridge's user only: the Matter fabric for which the bridge asks, as <code>fabricIndex</code> , <code>label</code> , and <code>vendorId</code> (Section 10.9).

Result	Type	Presence	Description
left	number (s)	OPTIONAL	How long until it ends. Absent when it has no end.
until	timestamp	OPTIONAL	When it ends. Absent when it has no end, and while the controller's clock is not valid.

automation.endScene: request, from an app or service. Ends the scene that a person started on a lighting function. Operation: `start-mode`. Audit record: `start-mode`, with the lighting function as `target.element`, and `{"end": true}` in `details`. Reasons it may give (Table 2): `bad-parameter` with `unknown-element`; `not-permitted` with `not-bridge`.

Parameter	Type	Presence	Description
function	identifier	REQUIRED	The lighting function.
fabric	object	OPTIONAL	From a Matter bridge's user only: the Matter fabric for which the bridge asks, as <code>fabricIndex</code> , <code>label</code> , and <code>vendorId</code> (Section 10.9).

Result: {}.

1. `automation.startScene` starts a scene as Lighting Programs Section 13.2 says that an owner starts one, for its own `duration` or for the shorter `duration` given, and the controller keeps it in the saved state [OAP-LP Section 7.7]. A controller MUST refuse a `duration` longer than the scene's own, with `bad-parameter` and the reason `duration-too-long`.
2. `function` names a lighting function, and `scene` a scene of its program; a controller refuses any other with the reason `unknown-element`.
3. `automation.endScene` MUST end the scene that a person started on the function, with its fade-out [OAP-LP Section 13.2.6]. A scene that a scene trigger started follows its condition, and `automation.endScene` does not end it. With no scene that a person started, it changes nothing, and succeeds.

10.5 Overrides

`automation.setOverride`: request, from an app or service. Sets an override of an output, replacing any override it has. Operation: `set-override`. Audit record: `set-override`, with the output as `target.element`, and `state`, `duration`, and `reason` in `details`. Reasons it may give (Table 2): `bad-parameter` with `unknown-element` and `invalid-state`; `bad-state` with `other-user`; `not-permitted` with `not-bridge`.

Parameter	Type	Presence	Description
output	identifier	REQUIRED	The output.
state	state	REQUIRED	Run, stop, or a level (Automation Table 4).
duration	number (s)	REQUIRED	How long the override lasts.
reason	string	OPTIONAL	The person's note.
onlyOwn	boolean	OPTIONAL	<code>true</code> to carry out the request only if no other user set the override or started the mode (Section 10.9). Default <code>false</code> .
fabric	object	OPTIONAL	From a Matter bridge's user only: the Matter fabric for which the bridge asks, as <code>fabricIndex</code> , <code>label</code> , and <code>vendorId</code> (Section 10.9).

Result	Type	Presence	Description
left	number (s)	REQUIRED	How long until the override ends.
until	timestamp	OPTIONAL	When it ends. Absent when it has no end, and while the controller's clock is not valid.

`automation.endOverride`: request, from an app or service. Ends the override of an output. Operation: `set-override`. Audit record: `set-override`, with the output as `target.element`, and `{"end": true}` in `details`. Reasons it may give (Table 2): `bad-parameter` with `unknown-element`; `bad-state` with `other-user`; `not-permitted` with `not-bridge`.

Parameter	Type	Presence	Description
output	identifier	REQUIRED	The output.
onlyOwn	boolean	OPTIONAL	<code>true</code> to carry out the request only if no other user set the override or started the mode (Section 10.9). Default <code>false</code> .
fabric	object	OPTIONAL	From a Matter bridge's user only: the Matter fabric for which the bridge asks, as <code>fabricIndex</code> , <code>label</code> , and <code>vendorId</code> (Section 10.9).

Result: {}.

1. `automation.setOverride` sets an override as Automation Section 15 defines it, which replaces any override of the output: it starts now, and expires after `duration`. A controller MUST record the session's user as the override's `by`.
2. A controller MUST refuse a `state` that the output cannot take, such as a level for an output whose class has none [OAP-AU Table 5], with `bad-parameter` and the reason `invalid-state`.

3. An app SHOULD propose an override of 1 hour, and SHOULD ask the person to confirm one of more than 24 hours [OAP-AU Section 15.1].
4. `automation.endOverride` ends the override of the output; with no override, it changes nothing, and succeeds.

The controller sends an override's commands with `revertAfter`, so that the override ends on the device even if the controller fails [OAP-AU Section 17.5].

10.6 Step Programs

automation.startProgram: request, from an app or service. Starts a step program, now or at a later time.

Operation: `start-program`. Audit record: `start-program`, with the program as `target.element`, and `start` in `details`. Reasons it may give (Table 2): `bad-parameter` with `unknown-element` and `start-passed`; `bad-state` with `program-conflict`, `clock-invalid`, and `unsupported`.

Parameter	Type	Presence	Description
<code>program</code>	identifier	REQUIRED	The step program.
<code>start</code>	timestamp	OPTIONAL	When the program starts. Default now.

Result	Type	Presence	Description
<code>start</code>	timestamp	REQUIRED	The program's start time.

automation.stopProgram: request, from an app or service. Stops a step program. Operation: `start-program`.

Audit record: `start-program`, with the program as `target.element`, and `{"end": true}` in `details`. Reasons it may give (Table 2): `bad-parameter` with `unknown-element`.

Parameter	Type	Presence	Description
<code>program</code>	identifier	REQUIRED	The step program.

Result: `{}`.

1. `automation.startProgram` starts a step program as Automation Section 16.3 says, now or at `start`.
2. A controller MUST refuse:
 - a `start` more than 60 s before its own time, with `bad-parameter` and the reason `start-passed`;
 - a program that runs already, or that sets a function or an output that another running program sets, with `bad-state` and the reason `program-conflict` [OAP-AU Section 16.5]; and
 - any `start` while its clock is not valid, with `bad-state` and the reason `clock-invalid`, because a program's position follows from its start time [OAP-AU Section 16.4.1].
3. `automation.stopProgram` stops the program as Automation Section 16.3.3 says, and cancels one that waits for its start time; with no such program, it changes nothing, and succeeds.

10.7 Latches

automation.clearLatch: request, from an app or service. Resets a latch, clears a latching interlock or a latched fault, or resumes or abandons an aborted water change. Operation: `clear-latch`. Audit record: `clear-latch`, with the element as `target.element`, or the device and the endpoint as `target.device` and `target.endpoint`, and `code` and `action` in `details`. Reasons it may give (Table 2): `bad-parameter` with `unknown-element` and `action-needed`; `bad-endpoint` with `unknown-device`; `bad-state` with `latch-set`, `owner-reset-off`, `not-latched`, `cannot-resume`, `device-lost`, and `device-refused`.

Parameter	Type	Presence	Description
<code>element</code>	identifier	One of	The latch condition, the latching interlock, or the function.
<code>device</code>	string	One of	The device whose endpoint has a fault that stays until it is cleared.
<code>endpoint</code>	string	With <code>device</code>	That endpoint.
<code>code</code>	string	OPTIONAL	The latched fault, for a function or endpoint that may have more than one, such as <code>max-on-time</code> or <code>testing-suspended</code> .
<code>action</code>	string	With <code>water-change-aborted</code>	<code>resume</code> or <code>abandon</code> an aborted water change (Water Level and Water Change Section 9.11).

Result: {}.

1. With `element`, `automation.clearLatch` resets a latch [OAP-AU Section 10.4], clears a latching interlock [OAP-AU Section 13.2], or clears a latched fault of a function, such as `max-on-time` [OAP-AU Section 11.8], `dry-run` or another latched fault of a top-off function [OAP-WL Section 7.6], `stuck-on` of a dosing head [OAP-DO Section 14.5.2], or `testing-suspended` [OAP-WT Section 10.5.5]. `code` names the fault when the function may have more than one.
2. A controller MUST refuse to reset a latch whose `set` condition is true or unknown, with `bad-state` and the reason `latch-set`, and one whose `ownerReset` is `false`, with the reason `owner-reset-off` [OAP-AU Section 10.4].
3. To clear `water-change-aborted`, a request MUST give `action: resume`, to go on to fill, or `abandon`, to return the water change to idle [OAP-WL Section 9.11.2]. A controller MUST refuse a request without it, with `bad-parameter` and the reason `action-needed`, and MUST refuse to resume a water change while a fill input is unavailable or there is a leak, with `bad-state` and the reason `cannot-resume`.
4. With `device` and `endpoint`, `automation.clearLatch` clears a fault that the endpoint keeps until it is cleared, such as a top-off unit's latched fault [OAP-AN Section 8.4.2]: the controller sends the endpoint `resetFault`, as its controller. A controller MUST refuse the request with `bad-state` and the reason `device-lost` while it has lost the device, and with the reason `device-refused`, and the device's error as `error`, when the device refuses the command.
5. Clearing what is not latched changes nothing, and succeeds, except that a request with `action` when no water change has aborted is refused with `bad-state` and the reason `not-latched`.

Clearing a latch does not by itself start anything: the function then runs as its inputs say [OAP-WL Section 7.6.2]. An app SHOULD ask the person to confirm that they have checked the system before it clears a latched fault [OAP-WL Section 7.6.4], and SHOULD warn, before an aborted water change is abandoned, that the top-off will replace the missing water with fresh water [OAP-WL Section 9.11.3].

10.8 Stopping Everything

automation.stopAll: request, from an app or service. Stops every output at once, as Security and Access Section 11.8 defines, or sets a new end for a `stop-all` in effect. Operation: `stop-all`. Audit record: `stop-all`, with `duration` in details.

Parameter	Type	Presence	Description
<code>duration</code>	number (s)	OPTIONAL	How long it lasts: from 1 s to 86 400 s. Default 3600 s.

Result	Type	Presence	Description
<code>left</code>	number (s)	REQUIRED	How long until it ends.
<code>until</code>	timestamp	OPTIONAL	When it ends. Absent when it has no end, and while the controller's clock is not valid.

automation.endStopAll: request, from an app or service. Ends `stop-all`. Operation: `stop-all`. Audit record: `stop-all`, with `{"end": true}` in details.

Parameters: none.

Result: {}.

1. `automation.stopAll` starts `stop-all` as Security and Access Section 11.8 defines it, for `duration`, at most 24 hours [OAP-SA Section 11.8.2]. While `stop-all` lasts, a new request gives it a new end, `duration` from now.
2. A controller MUST record the session's user as the user who started `stop-all` [OAP-AU Table 26].
3. A controller MUST start sending the stops of `stop-all` before it responds to `automation.stopAll`, and MUST NOT delay them for the response, the audit record, or any other request [OAP-SA Section 11.8.6].
4. A controller MUST carry out `automation.stopAll` and `automation.endStopAll` for every user whose role permits `stop-all`, in emergency sessions too, and while its clock is not valid [OAP-SA Section 11.8.4].
5. `automation.endStopAll` ends `stop-all` [OAP-SA Section 11.8.5]; when it is not in effect, it changes nothing, and succeeds.

10.9 Matter Bridges on Other Hosts

A Matter bridge on another host acts only through the controller, with the methods of this Specification, in a session as a party of kind `service` [OAP-MM Section 7.1.2]. It reads what it presents with `system.subscribe` and `automation.subscribe`, decides each Matter command from it as Matter Mapping Section 12.8 says, and carries it out as an override, a mode, or a scene, or, for a setpoint or a name, by replacing the configuration document [OAP-MM Sections 12.1, 12.7, and 13].

1. The bridge gives the fabric of each operation it asks for in `fabric` [OAP-MM Section 12.1.2]. A controller MUST record that request's audit record with `via matter`, and the fabric's `fabricIndex`, `label`, and `vendorId` in `details` [OAP-SA Section 12.1.6].
2. A controller MUST accept `fabric` only from a Matter bridge's user, and MUST refuse a request with `fabric` from any other user with `not-permitted` and the reason `not-bridge`.
3. With `onlyOwn true`, a controller MUST carry out `automation.setOverride` and `automation.endOverride` only when the output has no override, or one that the session's user set, and `automation.endMode` only when the mode is not active, or the session's user started it, and MUST refuse it otherwise with `bad-state` and the reason `other-user`. So a bridge never replaces or ends a person's override, nor ends a mode that a person started, even when the person acts between the bridge's reading of the state and its request [OAP-MM Sections 12.4.3, 12.6.2, and 13.1.7].
4. With `restart false`, a controller MUST leave a mode that is active as it is, without moving its end [OAP-MM Section 13.1.5].

This Specification has no method to rename an endpoint, so a bridge on another host renames the output, input, or equipment that a Matter endpoint presents instead [OAP-MM Table 8].

11 Alarms

11.1 Listing and Following Alarms

alarms.list: request, from an app or service. Returns every alarm and notice that is not closed. Operation: `view-state`.

Parameters: none.

Result	Type	Presence	Description
<code>alarms</code>	array	REQUIRED	Each alarm and notice that is not closed, as Alarms and Notifications Table 25 defines it.
<code>seq</code>	integer	REQUIRED	The <code>seq</code> of the latest alarm message that the controller had created (Section 11.1).

alarms.subscribe: request, from an app or service. Starts `alarms.changed` notifications on the session, and, unless `notify` is `false`, delivery of the app channel's messages for the session's user with `alarms.notify`.

Operation: `view-state`.

Parameter	Type	Presence	Description
<code>notify</code>	boolean	OPTIONAL	Whether the controller delivers the app channel's messages for the session's user on this session (Section 11.2). Default <code>true</code> .

Result: `{}`.

alarms.unsubscribe: request, from an app or service. Ends the session's subscription of the `alarms` namespace. Operation: `view-state`.

Parameters: none.

Result: `{}`.

alarms.changed: notification, from the controller. Reports a change of an alarm: an alarm message of the kind `raised`, `acknowledged`, `silenced`, `cleared`, or `closed`.

Parameter	Type	Presence	Description
<code>message</code>	object	REQUIRED	The alarm message (Alarms and Notifications Section 16.6).

1. `alarms.list` MUST return every alarm and notice that is not closed, with every member of Alarms and Notifications Table 25 that applies to it [OAP-AN Section 12.10.1].
2. After it responds to `alarms.subscribe`, a controller MUST send `alarms.changed` for every change of every alarm, as it reports them to a feed: `raised`, `acknowledged`, `silenced`, `cleared`, and `closed`, whatever its rules, policies, holds, and quiet hours say [OAP-AN Section 16.2.4]. Each change is one alarm message, with one `id` and one `seq`, which every subscribed session receives.
3. An app that subscribes, and then lists the alarms, applies only the messages of `alarms.changed` whose `seq` is greater than the `seq` of the list, so that it misses no change and applies none twice.

The history of closed alarms comes from `history.read` (Section 9.1.4).

11.2 Notifications Through Apps

`alarms.notify`: request, from the controller. Delivers a message of the `app` channel to the app of one of its recipients, which responds once it has received it.

Parameter	Type	Presence	Description
<code>message</code>	object	REQUIRED	The alarm message, with the <code>recipient</code> it is for.

Result: `{}`.

1. A session whose subscription of the `alarms` namespace has `notify true` is a session of its user for the `app` channel [OAP-AN Section 16.1.3]. A controller MUST send each message of the `app` channel for a recipient that is a user to each such session of that user, with `alarms.notify`.
2. An app MUST respond to `alarms.notify` as soon as it has received the message, before it shows it, so that the controller knows that the message reached the app.
3. A response with a result is delivery [OAP-AN Section 16.7.5]. For each app installation of the user, when no session of its credential responds within 10 s [OAP-DP Section 9.5.2], or it has no such session, a controller MUST send the message to the installation's push subscription, if it has one (Section 11.5) [OAP-AN Section 17.1.1].
4. A session in which a user signed in with a password belongs to no app installation: it receives `alarms.notify`, but has no push subscription.

An app shows each message as Alarms and Notifications Section 17.1.3 advises, with the most urgent class of notification its platform offers for a critical alarm.

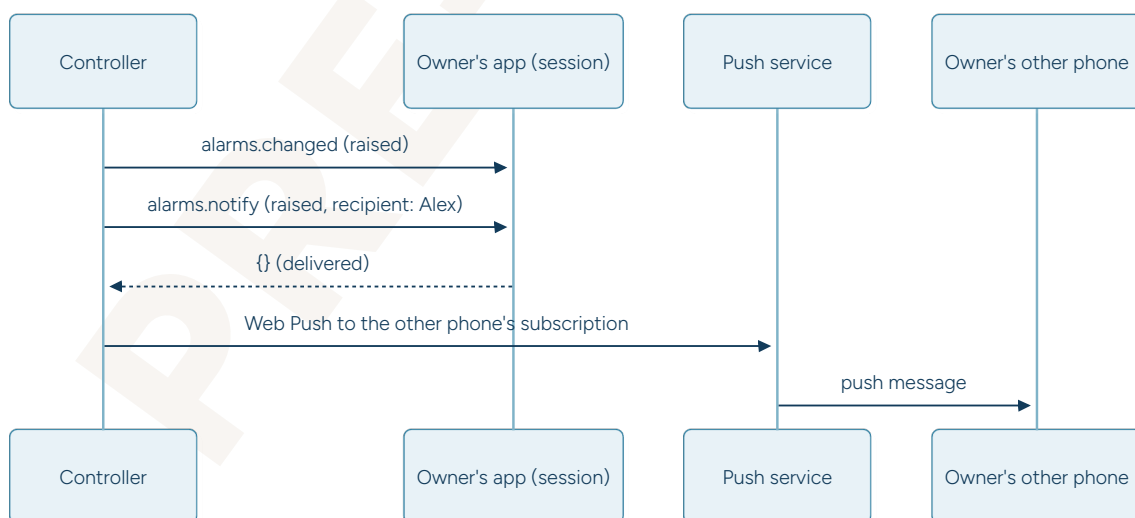


Figure 2 — A critical alarm reaches an owner's two phones

11.3 Acknowledging and Silencing

alarms.acknowledge: request, from an app or service. Acknowledges alarms (Alarms and Notifications Section 12.4). Operation: `acknowledge-alarm`. Audit record: `acknowledge-alarm`, with the `id` of each alarm acknowledged in `details`, as `alarms`. Reasons it may give (Table 2): `bad-parameter` with `unknown-alarm`.

Parameter	Type	Presence	Description
<code>alarms</code>	array	One of	The <code>id</code> of each alarm or notice to acknowledge.
<code>all</code>	true	One of	Acknowledge every alarm and notice that is not closed.

Result	Type	Presence	Description
<code>acknowledged</code>	array	REQUIRED	The <code>id</code> of each alarm and notice that this request acknowledged.

alarms.silence: request, from an app or service. Silences the alarms that are sounding (Alarms and Notifications Section 12.5). Operation: `acknowledge-alarm`. Audit record: `acknowledge-alarm`, with the `id` of each alarm silenced, and the silence's duration, in `details`, as `alarms` and `silence`.

Parameter	Type	Presence	Description
<code>duration</code>	number (s)	OPTIONAL	How long the silence lasts, from 60 s to 3600 s. Default the settings' <code>silenceDuration</code> .

Result	Type	Presence	Description
<code>alarms</code>	array	REQUIRED	The <code>id</code> of each alarm silenced.
<code>left</code>	number (s)	REQUIRED	How long until the silence ends.
<code>until</code>	timestamp	OPTIONAL	When it ends. Absent when it has no end, and while the controller's clock is not valid.

1. `alarms.acknowledge` acknowledges each alarm and notice listed, or, with `all`, each one that is not closed, as Alarms and Notifications Section 12.4 says. A controller MUST record the session's user as the one who acknowledged it [OAP-AN Section 12.4.4], and leaves an alarm that is acknowledged already as it is.
2. When an `id` names no alarm or notice that is not closed, a controller MUST refuse the request with `bad-parameter` and the reason `unknown-alarm`, and acknowledge none.
3. `alarms.silence` silences the alarms that are sounding, as Alarms and Notifications Section 12.5 says, for `duration`; it does not acknowledge them, and an alarm raised later sounds [OAP-AN Section 12.5.3]. With no alarm sounding, it silences nothing, and succeeds with an empty `alarms`.

11.4 Tests

alarms.test: request, from an app or service. Sends a test message to a recipient or a feed (Alarms and Notifications Section 16.8). Operation: `manage-consent`, or `manage-own-credentials` for a recipient that is the sender's own user. Audit record: `manage-consent`, or `manage-own-credentials`, as the operation it needed, with the recipient or the channel in `details`, as `recipient` or `channel`. Reasons it may give (Table 2): `bad-parameter` with `unknown-recipient`.

Parameter	Type	Presence	Description
<code>recipient</code>	identifier	One of	The recipient, which is sent a test through each of its contacts.
<code>channel</code>	identifier	One of	A feed: a webhook channel with <code>feed true</code> , or an MQTT channel.

Result	Type	Presence	Description
<code>message</code>	UUID	REQUIRED	The <code>id</code> of the test message.

1. `alarms.test` sends a test message to the recipient, through each of its contacts, or to the feed [OAP-AN Section 16.8].
2. A test of a recipient whose `user` is the session's user needs `manage-own-credentials`, and any other test `manage-consent` [OAP-AN Table 14]; a controller MUST check the operation that applies.
3. The result gives the test message's `id` as soon as the controller has created it; a channel that fails to deliver it raises `notification-failed`, as for any message [OAP-AN Section 16.7.3].

11.5 Push

alarms.readVapidKey: request, from an app or service. Returns the controller's VAPID public key, with which apps subscribe to push (Alarms and Notifications Section 17.2.2). Operation: [view-configuration](#).

Parameters: none.

Result	Type	Presence	Description
publicKey	string	REQUIRED	The public key, a P-256 point in uncompressed form, 65 bytes, in base64url.

alarms.registerPush: request, from an app or service. Registers the push subscription of the app installation whose credential opened the session, replacing any it had. Operation: [manage-own-credentials](#). Audit record: [manage-own-credentials](#), with the session's credential as `target.credential`, and `{"push": "registered"}` in `details`. Reasons it may give (Table 2): [bad-state](#) with [no-credential](#).

Parameter	Type	Presence	Description
endpoint	string	REQUIRED	The push service's URL for the subscription (RFC 8030).
p256dh	string	REQUIRED	The subscription's public key, a P-256 point in uncompressed form, 65 bytes, in base64url (RFC 8291).
auth	string	REQUIRED	The subscription's authentication secret, 16 bytes, in base64url (RFC 8291).

Result: `{}`.

alarms.unregisterPush: request, from an app or service. Removes the push subscription of the app installation whose credential opened the session. Operation: [manage-own-credentials](#). Audit record: [manage-own-credentials](#), with the session's credential as `target.credential`, and `{"push": "removed"}` in `details`. Reasons it may give (Table 2): [bad-state](#) with [no-credential](#).

Parameters: none.

Result: `{}`.

1. An app subscribes to its push service with the controller's VAPID public key, from [alarms.readVapidKey](#), and registers the subscription with [alarms.registerPush](#) [OAP-AN Section 17.2].
2. A controller MUST keep at most one push subscription for each credential: [alarms.registerPush](#) registers one for the credential that authenticated the session, in place of any earlier one, and a controller MUST refuse it, with [bad-state](#) and the reason [no-credential](#), in a session in which a user signed in with a password.
3. A controller MUST remove a credential's push subscription when the credential is revoked [OAP-AN Section 17.2.1].
4. Push subscriptions are not part of the alarm settings, and are not exported [OAP-AN Section 19.4.3].

11.6 Alarm Settings

alarms.readSettings: request, from an app or service. Returns the alarm settings document, without its secrets. Operation: [view-configuration](#).

Parameters: none.

Result	Type	Presence	Description
document	object	REQUIRED	The alarm settings document (Alarms and Notifications Section 19), without passwords and webhook secrets.
version	string	REQUIRED	The document's version. A string of 1 to 64 characters that changes whenever the document changes (Section 7.6).
secrets	array	REQUIRED	The <code>id</code> of each channel whose password or webhook secret the controller holds.

alarms.validateSettings: request, from an app or service. Checks an alarm settings document as [alarms.replaceSettings](#) would, without changing anything. Operation: [view-configuration](#).

Parameter	Type	Presence	Description
document	object	REQUIRED	The alarm settings document to check.

Result	Type	Presence	Description
valid	boolean	REQUIRED	Whether the controller would accept the document.
errors	array	REQUIRED	Why it would not, as for <code>automation.validateConfiguration</code> .
warnings	array	REQUIRED	What the controller would warn the owner of.

`alarms.replaceSettings`: request, from an app or service. Replaces the alarm settings document. Operation: `change-configuration`, or `manage-consent` or `manage-own-credentials`, each as the changes need (Section 11.6). Audit record: `change-configuration`, or `manage-consent` or `manage-own-credentials`, one record for each operation it needed, with the previous and the new `version` in `details`, as `from` and `to`, and the top-level members that changed, as `changed`. Reasons it may give (Table 2): `bad-parameter` with `invalid-document`; `bad-state` with `version-conflict`.

Parameter	Type	Presence	Description
document	object	REQUIRED	The new alarm settings document.
ifVersion	string	OPTIONAL	Replace it only if the current document has this version.

Result	Type	Presence	Description
version	string	REQUIRED	The new document's version.
secrets	array	REQUIRED	The webhook secret that the controller made for each new webhook channel without one, as its <code>channel</code> and <code>secret</code> , shown only this once (Alarms and Notifications Section 17.4.4).
warnings	array	REQUIRED	What the controller warns the owner of.

1. `alarms.readSettings` MUST return the complete alarm settings document, without passwords and webhook secrets [OAP-AN Section 19.4.1], and, in `secrets`, each channel whose password or secret the controller holds.
2. `alarms.validateSettings` MUST check a document as Alarms and Notifications Section 19.5.2 requires, and MUST return as warnings what that specification requires a controller to warn the owner of, such as a critical alarm that notifies nobody [OAP-AN Section 16.4.3], and a reference that it cannot resolve [OAP-AN Section 19.3.3].
3. `alarms.replaceSettings` needs, for each change between the current document and the new one, the operation of Table 6 [OAP-AN Table 14], and a controller MUST refuse it, with `not-permitted`, when the user's role does not permit each of them.
4. A controller MUST refuse, with `bad-parameter` and the reason `invalid-document`, an `alarms.replaceSettings` whose document it would not accept, with the problems in `errors`, and MUST then keep its settings [OAP-AN Section 19.5.2].
5. A channel of the new document without a password or a secret keeps that of the current document's channel with the same `id` and `type`. For each new webhook channel without a secret, a controller MUST make a secret [OAP-AN Section 17.4.4], and MUST return it in that result's `secrets`, and in no other result.

Table 6 — What each change of alarm settings needs

Change	Operation
The quiet hours of the recipient whose <code>user</code> is the session's user, and nothing else of it	<code>manage-own-credentials</code>
Any other change of <code>recipients</code> , and any change of <code>channels</code> or <code>heartbeats</code>	<code>manage-consent</code>
Any change of another top-level member, such as <code>ranges</code> , <code>powerChecks</code> , <code>customAlarms</code> , <code>conditions</code> , <code>local</code> , <code>policies</code> , or <code>rules</code>	<code>change-configuration</code>

So a house-sitter with the `view` role can set their own quiet hours, an owner decides where alarms leave the owner's network, and a user with the `full-control` role changes ranges and rules [OAP-AN Section 12.10].

12 Users and Sharing

12.1 Users

users.list: request, from an app or service. Returns the controller's users. Operation: `view-access`; a user without it receives only their own entry.

Parameters: none.

Result	Type	Presence	Description
users	array	REQUIRED	Each user (Table 7), or, for a user without <code>view-access</code> , only that user.

1. `users.list` MUST return every user to a user whose role permits `view-access`, and only the session's own user to any other, so that every user can find their own credentials to revoke one [OAP-SA Section 11.7.1].
2. It MUST NOT return any secret, such as a password hash, a verifier, or a code [OAP-SA Section 8.4].

Table 7 — Members of a user

Member	Type	Presence	Description
id	UUID	REQUIRED	The user's id [OAP-SA Section 11.1].
name	string	REQUIRED	The user's name.
kind	string	REQUIRED	<code>person</code> or <code>service</code> .
role	string	REQUIRED	The user's role.
grantExpires	timestamp	OPTIONAL	When the user's grant ends.
expired	boolean	OPTIONAL	<code>true</code> when the grant has ended, and the controller keeps the user without access, so that an owner can renew it [OAP-SA Section 11.4.2].
remote	boolean	REQUIRED	Whether the user may connect remotely.
created, createdBy	timestamp, UUID	OPTIONAL	When, and by which user, the user was created.
credentials	array	REQUIRED	The user's app installations: each a <code>keyId</code> , its <code>label</code> , when it was <code>added</code> , or <code>null</code> , and <code>current</code> <code>true</code> for the one that authenticated the session.
password	boolean	REQUIRED	Whether the user has a password.
extensions	object	OPTIONAL	Extensions.

12.2 Invitations

A user invites a new user, or adds a new app of their own, with the Device Protocol's `pairing.invite` [OAP-DP Section 10.5], which needs `manage-users`, or `manage-own-credentials` with `self` [OAP-SA Section 11.5], and which a controller serves [OAP-DP Section 17.2]. This Specification defines no other method for it. The invited person's app pairs with the invitation, on the local network or through a relay, and the new user then appears in `users.list`.

12.3 Changing and Removing Users

users.update: request, from an app or service. Changes a user's name, role, expiry, or remote access.

Operation: `manage-users`. Audit record: `manage-users`, with the user as `target.user`, and each change in details, as changes, each with its `member` and its values `from` and `to`. Reasons it may give (Table 2): `bad-parameter` with `unknown-user`, `service-owner`, and `owner-expiry`; `bad-state` with `last-owner`.

Parameter	Type	Presence	Description
user	UUID	REQUIRED	The user's id.
name	string	OPTIONAL	The new name.
role	string	OPTIONAL	The new role.
grantExpires	timestamp or null	OPTIONAL	When the grant ends, or <code>null</code> for a grant that does not end.
remote	boolean	OPTIONAL	Whether the user may connect remotely.

Result	Type	Presence	Description
user	object	REQUIRED	The user, as it is now (Table 7).

users.remove: request, from an app or service. Removes a user, and revokes every credential of the user.

Operation: `manage-users`. Audit record: `manage-users`, with the user as `target.user`, and `{"removed": true}` in `details`. Reasons it may give (Table 2): `bad-parameter` with `unknown-user`; `bad-state` with `last-owner`.

Parameter	Type	Presence	Description
<code>user</code>	UUID	REQUIRED	The user's id.

Result: `{}`.

users.revokeCredential: request, from an app or service. Revokes one credential of a user, such as a lost phone's. Operation: `manage-users`, or `manage-own-credentials` for a credential of the sender's own user. Audit record: `manage-users`, or `manage-own-credentials`, as the operation it needed, with the key ID as `target.credential`, and its user as `target.user`. Reasons it may give (Table 2): `bad-parameter` with `unknown-credential`; `bad-state` with `no-credential`.

Parameter	Type	Presence	Description
<code>keyId</code>	string	OPTIONAL	The credential's key ID. Default the credential that opened the session.

Result: `{}`.

- `users.update` changes the members it gives. A controller MUST refuse:
 - a change that would leave it without an owner, with `bad-state` and the reason `last-owner` [OAP-SA Section 11.1.4];
 - the `owner` role for a user of kind `service`, with `bad-parameter` and the reason `service-owner` [OAP-SA Section 11.1.3]; and
 - an expiry for a user with the `owner` role, with `bad-parameter` and the reason `owner-expiry` [OAP-SA Section 11.4.1].
- `users.remove` removes the user, and revokes each of its credentials [OAP-SA Section 11.7.1]. A controller MUST refuse to remove its last owner, with `bad-state` and the reason `last-owner`.
- When it revokes a credential, for `users.remove` or `users.revokeCredential`, a controller MUST close every session authenticated with it within 5 s, with the code 4001 [OAP-DP Section 10.1.3], MUST remove its push subscription (Section 11.5.3), and, for a credential of an owner, MUST remove its key from the pairing lists of the system's devices [OAP-SA Section 11.7.2]; and when it removes a user, it MUST also close every session in which the user signed in with a password.
- `users.revokeCredential` needs `manage-own-credentials` for a credential of the session's own user, and `manage-users` for any other [OAP-SA Section 11.7.1]. Without `keyId`, it revokes the credential that authenticated the session, and a controller refuses it, with `bad-state` and the reason `no-credential`, in a session in which a user signed in with a password.

Overrides and modes that a removed user started do not end because the user was removed [OAP-SA Section 11.4.3]. An app SHOULD ask the person to confirm again before `users.update` and `users.remove` [OAP-SA Section 11.3.6].

12.4 The Audit Log

users.readAudit: request, from an app or service. Returns records of the controller's audit log, in order.

Operation: `view-audit-log`.

Parameter	Type	Presence	Description
<code>after</code>	integer	OPTIONAL	Return the records whose <code>seq</code> is greater. Default 0.
<code>limit</code>	integer	OPTIONAL	The most records to return, from 1 to 1000. Default 100.

Result	Type	Presence	Description
<code>records</code>	array	REQUIRED	The records, in the form of Security and Access Table 7, in order of <code>seq</code> .
<code>more</code>	boolean	REQUIRED	<code>true</code> when more records follow.

- `users.readAudit` MUST return the records whose `seq` is greater than `after`, in order of `seq`, at most `limit` of them, in the form of Security and Access Table 7, and `more` `true` when more records follow.

2. An app that reads the whole log asks again with `after` set to the last `seq` it received, until `more` is `false`. A gap in `seq` is a record that the controller dropped or lost [OAP-SA Section 12.2].

The audit log is exported as a document in an export file (Section 14.1) [OAP-SA Section 12.3.3].

12.5 Passwords and Labels

users.setPassword: request, from an app or service. Sets the password of the session's own user, for the web interface and for services that sign in (Security and Access Section 10.5). Operation: `manage-own-credentials`. Audit record: `manage-own-credentials`, with the user as `target.user`, and `{"password": "set"}` in `details`; never the password. Reasons it may give (Table 2): `bad-parameter` with `weak-password`.

Parameter	Type	Presence	Description
<code>password</code>	string	REQUIRED	The new password, which the user chose.

Result: `{}`.

users.removePassword: request, from an app or service. Removes a user's password. Operation: `manage-users`, or `manage-own-credentials` for the session's own user. Audit record: `manage-users`, or `manage-own-credentials`, as the operation it needed, with the user as `target.user`, and `{"password": "removed"}` in `details`. Reasons it may give (Table 2): `bad-parameter` with `unknown-user`.

Parameter	Type	Presence	Description
<code>user</code>	UUID	OPTIONAL	The user's id. Default the session's own user.

Result: `{}`.

users.labelCredential: request, from an app or service. Changes the label of a credential, such as "Alex's phone". Operation: `manage-users`, or `manage-own-credentials` for a credential of the session's own user. Audit record: `manage-users`, or `manage-own-credentials`, as the operation it needed, with the key ID as `target.credential`, and the new `label` in `details`. Reasons it may give (Table 2): `bad-parameter` with `unknown-credential`; `bad-state` with `no-credential`.

Parameter	Type	Presence	Description
<code>keyId</code>	string	OPTIONAL	The credential's key ID. Default the credential that opened the session.
<code>label</code>	string	REQUIRED	The new label.

Result: `{}`.

1. `users.setPassword` sets the password of the session's own user, which that user chose; no user sets another's [OAP-SA Section 10.5.1]. A controller MUST refuse a password shorter than 15 characters, or on its list of common and compromised passwords, with `bad-parameter` and the reason `weak-password`, MUST accept passwords of at least 64 characters, and MUST impose no other rule on them [OAP-SA Section 10.5.1].
2. A controller MUST keep only a salted hash of the password [OAP-SA Section 10.5.2], and MUST NOT return it in any result.
3. `users.removePassword` removes a user's password: the session's own user's with `manage-own-credentials`, and another's with `manage-users` [OAP-SA Section 11.7.1]. A controller MUST close every session in which the user signed in with the password within 5 s [OAP-SA Section 10.6.1].
4. `users.labelCredential` changes the label of a credential: one of the session's own user with `manage-own-credentials`, and any other with `manage-users`.

13 Records

13.1 Maintenance

records.recordMaintenance: request, from an app or service. Records maintenance that changes no output: a refill, a replaced reagent, an emptied waste container, or other maintenance. Operation: `record-maintenance`. Audit record: `record-maintenance`, with the equipment or the vessel as `target.element`, and `kind`, and `channel`

or reagent where given, in details. Reasons it may give (Table 2): bad-parameter with unknown-element and invalid-record; bad-state with device-lost and device-refused.

Parameter	Type	Presence	Description
kind	string	REQUIRED	refill, reagent, waste, or maintenance (Table 8).
equipment	identifier	See Table 8	The equipment: the doser of a container, the analyzer, or the equipment maintained.
channel	identifier	See Table 8	The head of a doser, as a Channel of equipment.
vessel	identifier	See Table 8	The reservoir refilled.
reagent	identifier	See Table 8	The analyzer's reagent replaced.
volume	Quantity	OPTIONAL	For a refill, the volume in the container or reservoir after it. Default its capacity.
left	Quantity	OPTIONAL	For a refill of a doser's container, the volume that was left before it (Dosing Section 11.4.2).
expires	date	OPTIONAL	For a refill or a reagent, the date the product or the reagent expires.
lot	string	OPTIONAL	For a reagent, its lot.
time	timestamp	OPTIONAL	When it was done. Default now.
task	identifier	OPTIONAL	The Data Model Task that it completes.
notes	string	OPTIONAL	Free text.

Result	Type	Presence	Description
events	array	REQUIRED	The Data Model events that the controller recorded for it.

1. `records.recordMaintenance` records maintenance that changes no output [OAP-SA Table 6], of one of the kinds of Table 8, with the members that Table 8 gives for it. A controller MUST refuse other members, with bad-parameter and the reason invalid-record, and equipment, a head, a vessel, or a reagent that is not of the kind, with the reason unknown-element.
2. For each kind, a controller MUST do what Table 8 says, as the specification of the equipment says a person's record does, and MUST record the events that Table 8 gives.
3. With task, the event completes the task [OAP-DM Section 8.11], and, for the kind maintenance, has the task's eventType.
4. Where Table 8 has the controller send a command to the equipment, a controller MUST refuse the request with bad-state and the reason device-lost while it has lost the device, and with the reason device-refused, and the device's error as error, when the device refuses the command.

Table 8 — Kinds of maintenance record

kind	Members	What the controller does	What it records
refill, of a doser's container	equipment and channel, REQUIRED: the doser and its head; volume, left, and expires, OPTIONAL	Resets the head's estimate to volume, or its capacity, and clears low, expired, and, unless its condition is still true, empty [OAP-DO Section 11.4].	An event of the type refill, as Dosing Section 16.3.2 defines it.
refill, of a reservoir	vessel, REQUIRED; volume, OPTIONAL	Resets the top-off's estimate of the reservoir to volume, or its capacity [OAP-WL Section 8.12.6].	An event of the type refill, with the reservoir as vessel [OAP-WL Section 9.15].
reagent	equipment and reagent, REQUIRED: the analyzer and its reagent; expires and lot, OPTIONAL	Records the replacement with the analyzer's replaceReagent command, or, for a reagent that it tracks, by itself, which resets the estimate and clears low, empty, and expired [OAP-WT Section 9.6].	An event of the type maintenance, as Water Testing Section 15.4 defines it.
waste	equipment, REQUIRED: the analyzer	Records the emptying with the analyzer's emptyWaste command, or by itself [OAP-WT Section 9.6.4].	An event of the type maintenance, as Water Testing Section 15.4 defines it.
maintenance	equipment, OPTIONAL	Nothing more.	An event of the type maintenance, or of the task's eventType, with equipment.

time, task, and notes go with every kind.

13.2 Manual Results

records.enterResults: request, from an app or service. Enters the results of manual tests (Water Testing Section 12.3). Operation: **enter-results**. Audit record: **enter-results**, with the number of results recorded, and their parameters, in **details**, as **count** and **parameters**; a check is not recorded. Reasons it may give (Table 2): **bad-parameter** with **invalid-result**.

Parameter	Type	Presence	Description
results	array	REQUIRED	The results, from 1 to 100, each as Table 9 defines.
check	boolean	OPTIONAL	true to check the results without recording them. Default false .

Result	Type	Presence	Description
results	array	REQUIRED	What became of each result, in order (Table 10).

1. **records.enterResults** enters the results of manual tests as Water Testing Section 12.3 says, each with the members of Table 9.
2. A controller MUST refuse the whole request, with **bad-parameter** and the reason **invalid-result**, when a result's unit is not accepted for its parameter [OAP-DM Section 9.2], or its system or vessel is not one of its own.
3. A controller checks each result as Water Testing Section 12.3.4 says, and MUST NOT record a result that fails the check unless it has **confirmed true**. The outcome of such a result is **needs-confirmation**, with its **reasons**: **range** when it is outside the plausible range, and **jump** when it moved further than **maxJump**. The app asks the person, and, if they confirm it, sends it again with **confirmed true**.
4. A controller MUST record either every result of a request or none: when a result's outcome is **needs-confirmation**, it records none, and the outcome of each other result is **checked**.
5. With **check true**, a controller records nothing, and the outcome of each result that it would record is **checked**.
6. Each outcome that is **recorded** or **checked** has the **observation** that the controller records, or would record [OAP-WT Section 15.1], and **converted**, the value in the unit of the parameter's entry, which an app SHOULD show the person before it sends the result to be recorded [OAP-WT Section 12.3.2].

Table 9 — Members of a manual result

Member	Type	Presence	Description
parameter	string	REQUIRED	The Data Model parameter.
value	number	REQUIRED	The value, in unit .
unit	string	REQUIRED	The unit, as the instrument reported it, with its chemical basis [OAP-DM Section 9.4].
qualifier	string	OPTIONAL	< or >, for a value beyond the test's range [OAP-DM Section 8.4].
unitAssumed	boolean	OPTIONAL	true when the person could not give the chemical basis [OAP-WT Section 12.3.3]. Default false .
system, vessel	identifier	system REQUIRED	Where the water was taken from.
method	Method	REQUIRED	How it was measured: a Data Model Method, with the test kit or instrument as instrument [OAP-DM Section 8.4].
time	timestamp	OPTIONAL	When the sample was taken. Default the time of entry.
notes	string	OPTIONAL	Free text.
confirmed	boolean	OPTIONAL	true when the person confirmed a result that failed its check. Default false .
extensions	object	OPTIONAL	Extensions.

Table 10 — What becomes of a result

status	Meaning
recorded	The controller recorded it.

status	Meaning
checked	The controller would record it, but recorded nothing: the request was a check, or another result needs confirmation.
needs-confirmation	It failed its check, and the controller recorded nothing; <code>reasons</code> says why.
duplicate	From <code>records.importResults</code> only: the controller has imported its sample already, and records nothing of it.

13.3 Laboratory Results

`records.importResults`: request, from an app or service. Imports the results of a laboratory, as a Data Model Document (Water Testing Section 12.4). Operation: `enter-results`. Audit record: `enter-results`, with the identifier of each sample imported, and its laboratory's `reference` where it has one, in `details`, as `samples`; a check is not recorded. Reasons it may give (Table 2): `bad-parameter` with `invalid-document`, `not-laboratory`, and `unknown-element`.

Parameter	Type	Presence	Description
<code>document</code>	object	REQUIRED	The laboratory's Data Model Document.
<code>systems</code>	object	OPTIONAL	The controller's system for each system of the document, by the document's identifier. Default the system of the same <code>id</code> .
<code>vessels</code>	object	OPTIONAL	The controller's vessel for each vessel of the document, by the document's identifier. Default the vessel of the same <code>id</code> ; a vessel with neither is left out, so that its results describe the system's circulating water.
<code>confirmed</code>	array	OPTIONAL	The index, in the document's <code>observations</code> , of each result outside its plausible range that the person confirmed.
<code>check</code>	boolean	OPTIONAL	<code>true</code> to check the document without recording it. Default <code>false</code> .

Result	Type	Presence	Description
<code>observations</code>	array	REQUIRED	What became of each observation of the document, in order: its <code>status</code> , and <code>reasons</code> (Table 10).

1. `records.importResults` imports a laboratory's results, as a Data Model Document, as Water Testing Section 12.4 says: the Document's observations, and the samples they belong to.
2. A controller MUST refuse, with `bad-parameter`, a document that is not a valid Data Model Document, with the reason `invalid-document`, and one with an observation that does not belong to a Sample with `laboratory`, with the reason `not-laboratory`.
3. `systems` gives the controller's system for each system of the document, which the person chooses [OAP-WT Section 12.4.1], and `vessels` its vessel for each vessel of the document; a system of the document without one, and with no system of the controller of the same `id`, is refused with the reason `unknown-element`, and a vessel without one, and with no vessel of the same `id`, is left out, so that its results describe the system's circulating water [OAP-DM Section 8.4].
4. A controller MUST NOT record an observation outside its parameter's plausible range unless its index is in `confirmed` [OAP-WT Section 12.4.2], and gives its outcome as `needs-confirmation`, with the reason `range`.
5. A controller MUST NOT import a sample that it has imported already [OAP-WT Section 12.4.5]: the outcome of each observation of such a sample is `duplicate`, so that the app can tell the person.
6. A controller MUST record either every observation of a request that is not `duplicate`, or none, as Section 13.2.4 says, and, with `check true`, records nothing, as Section 13.2.5 says.

14 Export and Import

14.1 Export

`export.create`: request, from an app or service. Makes an export file (Data Model Section 7.8), and offers it to the session. Operation: `export-data`. Audit record: `export-data`, with `from` and `to`, and the `kind` of each member of the file, in `details`, as `from`, `to`, and `members`.

Parameter	Type	Presence	Description
from	timestamp	OPTIONAL	The start of the history to include. Default the oldest the controller holds.
to	timestamp	OPTIONAL	The end of the history to include. Default now.
auditLog	boolean	OPTIONAL	Whether to include the audit log, for a user allowed <code>view-audit-log</code> . Default <code>true</code> .

Result	Type	Presence	Description
file	string	REQUIRED	The file's name, for <code>export.read</code> .
size	integer	REQUIRED	Its size, in bytes.
sha256	string	REQUIRED	Its SHA-256 hash, in lowercase hexadecimal.
manifest	object	REQUIRED	Its manifest (Data Model Section 7.8.3).

export.read: request, from an app or service. Reads part of an export file that the controller offered to the session. Operation: `export-data`. Audit record: none of its own; `export.create` records the export. Reasons it may give (Table 2): `bad-parameter` with `unknown-file`.

Parameter	Type	Presence	Description
file	string	REQUIRED	The file's name.
offset	integer	REQUIRED	Where to start, in bytes.
length	integer	REQUIRED	How many bytes to read.

Result	Type	Presence	Description
data	string	REQUIRED	The bytes, in base64url.
eof	boolean	REQUIRED	<code>true</code> when the data reaches the end of the file.

1. `export.create` MUST make an export file as Data Model Section 7.8 defines it, with a Document of the history from `from` to `to`, the configuration document with its saved state, and the alarm settings document, and, when `auditLog` is `true` and the user's role permits `view-audit-log`, the audit log [OAP-DM Sections 7.8.6 and 7.8.7; OAP-SA Section 16.6.1].
2. A controller offers the file to the session that asked for it: it MUST serve `export.read` for the file on that session, and on no other, until the session ends, or until 10 minutes (provisional) have passed since the last request for the file, whichever is first.
3. The app chooses `length` so that the response fits in the largest message it accepts, and the controller MAY return fewer bytes [OAP-DP Section 13.2.3]. An app checks the file it has read against `size` and `sha256`.

The file holds no secrets: alarm settings are exported without them [OAP-AN Section 19.4.1]. Its Document holds the alarms that are not closed as well as those in history, as Alarms and Notifications Section 20.1.1 advises.

14.2 Import

export.upload: request, from an app or service. Sends the controller part of an export file to import. Operation: `change-configuration` or `enter-results`. Audit record: none of its own; `export.import` records the import. Reasons it may give (Table 2): `bad-parameter` with `unknown-file`, `file-too-large`, and `out-of-order`.

Parameter	Type	Presence	Description
file	string	After the first part	The file's name, from the result of its first part.
offset	integer	REQUIRED	Where the part goes, in bytes: the number of bytes received so far.
data	string	REQUIRED	The bytes, in base64url.
size	integer	In the first part	The size of the whole file, in bytes.
sha256	string	OPTIONAL	In the first part, the whole file's SHA-256 hash, in lowercase hexadecimal.

Result	Type	Presence	Description
file	string	REQUIRED	The file's name.
received	integer	REQUIRED	The number of bytes received so far.

export.import: request, from an app or service. Checks an export file that the session uploaded, or imports it. Operation: `change-configuration`, or `manage-consent` or `enter-results`, each as the members imported need (Section 14.2). Audit record: `change-configuration`, or `manage-consent` or `enter-results`, one record for each member imported and each operation it needed, with the member's `path` and `kind`, and, for a configuration document or alarm settings, the previous and the new `version`, in `details`; a check is not recorded. Reasons it may give (Table 2): `bad-parameter` with `unknown-file`, `invalid-export`, and `invalid-document`; `bad-state` with `incomplete`.

Parameter	Type	Presence	Description
<code>file</code>	string	REQUIRED	The file's name.
<code>members</code>	array	OPTIONAL	The <code>path</code> of each member to import. Default every member of a kind that the controller imports.
<code>check</code>	boolean	OPTIONAL	<code>true</code> to check the file without importing it. Default <code>false</code> .

Result	Type	Presence	Description
<code>members</code>	array	REQUIRED	What became of each member of the file (Table 11).

1. An app sends an export file to a controller in parts, with `export.upload`: the first part without `file`, and with `size`; each later part with the `file` that the first result gave, at the `offset` of the bytes received so far. A controller MUST refuse a part at any other offset, with `bad-parameter` and the reason `out-of-order`.
2. A controller MUST accept files of at least 4 194 304 bytes (provisional), and MUST refuse, in its first part, a file larger than it can hold, with `bad-parameter`, the reason `file-too-large`, and the largest it can hold as `limit`.
3. A controller keeps an uploaded file for the session, as Section 14.1.2 says of an export, and refuses `export.import` of a file whose bytes it has not all received, or whose SHA-256 hash is not the `sha256` of its first part, with `bad-state` and the reason `incomplete`.
4. `export.import` checks the file as Data Model Section 7.8.8 says, and each member that it would import as that member's specification says: the configuration document as Automation Section 19.2 does, the alarm settings as Alarms and Notifications Section 19.5 does, and the Document as Data Model Section 10.2 does. A controller MUST refuse a file that breaks Data Model Section 7.8, items 1 to 5, with `bad-parameter` and the reason `invalid-export`, and a file with a member to import that is not valid, with the reason `invalid-document`, and MUST then import nothing.
5. A controller MUST check the size of each member before it decompresses it, and MUST refuse, with the reason `file-too-large`, a file whose members would decompress to more than it can hold.
6. The members to import are those named in `members`, or, without `members`, every member of a kind that the controller imports. Importing them needs the operations of Table 12, and a controller MUST refuse the import, with `not-permitted`, when the user's role does not permit each operation that the members to import need.
7. A controller MUST import the members all or none, and each as its specification says:
 - the configuration document, with its saved state, in place of the running one [OAP-AU Section 19.2.6], with a new version;
 - the alarm settings, in place of the current ones, with a new version, keeping the secrets of channels of the same `id` and `type`, as Section 11.6.5 says; and
 - the records of the Document, added to the history, without recording twice an object whose `id` the history holds already [OAP-DM Section 7.3.3].
8. A controller MUST NOT import the audit log of an export file, nor add its records to its own audit log, whose records come only from what it did itself [OAP-SA Section 12.3.2]; the status of that member is `ignored`, as is that of a member of a kind that the controller does not import.
9. With `check true`, a controller imports nothing, and gives each member its outcome, with the operations it would need, its problems, and its warnings, as Section 7.6 says, so that the app can show the person what the import would do (Section 7.6.6).

Table 11 — What becomes of a member

Member	Type	Presence	Description
path	string	REQUIRED	The member's path in the file.
kind	string	REQUIRED	Its <code>kind</code> in the manifest [OAP-DM Section 7.8.4].
revision	string	OPTIONAL	Its revision in the manifest.
status	string	REQUIRED	<code>imported</code> ; <code>checked</code> , for a member that a check would import; <code>ignored</code> , for a member that is not imported; or <code>invalid</code> , for a member that is not valid.
operations	array	OPTIONAL	The operations that importing it needs (Table 12).
errors	array	OPTIONAL	Why it is not valid (Section 7.6.4).
warnings	array	OPTIONAL	What the controller would warn the owner of (Section 7.6.5).
version	string	OPTIONAL	For an imported configuration document or alarm settings, its new version.
extensions	object	OPTIONAL	Extensions.

Table 12 — What importing each kind of member needs

kind	Operations
automation	<code>change-configuration</code> [OAP-SA Table 6].
alarm-settings	<code>change-configuration</code> , and also <code>manage-consent</code> when it has <code>recipients</code> , <code>channels</code> , or <code>heartbeats</code> [OAP-AN Table 14].
data-model	<code>enter-results</code> , because its results can change how much a dosing function adds [OAP-SA Table 6].
audit-log	None: it is never imported (item 8).

15 Actions

15.1 Actions on Demand

actions.feed: request, from an app or service. Feeds now, with a feeder's `feed` command (Device Classes Section 9.9). Operation: `run-action`. Audit record: `run-action`, with the device and the endpoint as `target.device` and `target.endpoint`, and `{"feed": portions}` in `details`. Reasons it may give (Table 2): `bad-endpoint` with `unknown-device`; `bad-command` with `wrong-class`; `bad-state` with `held`, `device-lost`, `device-refused`, and `device-unanswered`; `local-control` with `under-local-control`.

Parameter	Type	Presence	Description
device	string	REQUIRED	The device, as Automation Section 8.5 identifies it.
endpoint	string	REQUIRED	The endpoint.
portions	integer	OPTIONAL	How many portions, from 1 to 10. Default 1.

Result: `{}`.

actions.test: request, from an app or service. Starts a test on demand, or a test of a standard, with an analyzer's `test` command (Water Testing Section 8.8). Operation: `run-action`. Audit record: `run-action`, with the device and the endpoint as `target.device` and `target.endpoint`, and `{"test": parameters}`, with `standard` when it is `true`, in `details`. Reasons it may give (Table 2): `bad-endpoint` with `unknown-device`; `bad-command` with `wrong-class`; `bad-state` with `held`, `device-lost`, `device-refused`, `device-unanswered`, and `clock-invalid`; `local-control` with `under-local-control`.

Parameter	Type	Presence	Description
device	string	REQUIRED	The device, as Automation Section 8.5 identifies it.
endpoint	string	REQUIRED	The endpoint.
parameters	array	OPTIONAL	The parameters to test, each one that the analyzer measures. Default all.
standard	boolean	OPTIONAL	<code>true</code> for a test of a standard rather than of the system's water. Default <code>false</code> .

Result: `{}`.

actions.fill: request, from an app or service. Starts a manual fill of a top-off unit, with its `fill` command (Water Level and Water Change Section 8.14). Operation: `run-action`, or `dose-now` for a unit whose water

carries an additive. Audit record: `run-action`, or `dose-now` for a unit whose water carries an additive, with the device and the endpoint as `target.device` and `target.endpoint`, and `{"fill": duration}` in `details`. Reasons it may give (Table 2): `bad-endpoint` with `unknown-device`; `bad-command` with `wrong-class`; `bad-state` with `held`, `device-lost`, `device-refused`, `device-unanswered`, and `limit-reached`; `local-control` with `under-local-control`.

Parameter	Type	Presence	Description
<code>device</code>	string	REQUIRED	The device, as Automation Section 8.5 identifies it.
<code>endpoint</code>	string	REQUIRED	The endpoint.
<code>duration</code>	number (s)	REQUIRED	The longest the fill may run; it still stops at the high sensor and at <code>maxFillTime</code> .

Result: `{}`.

actions.dose: request, from an app or service. Gives a manual dose, forward or in reverse, with a dosing head's dose command (Dosing Section 10.6). Operation: `dose-now`. Audit record: `dose-now`, with the device and the endpoint as `target.device` and `target.endpoint`, and `volume`, `rate`, and `direction` in `details`; a check is not recorded. Reasons it may give (Table 2): `bad-endpoint` with `unknown-device`; `bad-command` with `wrong-class`; `bad-state` with `held`, `device-lost`, `device-refused`, and `device-unanswered`; `local-control` with `under-local-control`.

Parameter	Type	Presence	Description
<code>device</code>	string	REQUIRED	The device, as Automation Section 8.5 identifies it.
<code>endpoint</code>	string	REQUIRED	The endpoint.
<code>volume</code>	number (mL)	REQUIRED	The volume.
<code>rate</code>	number (mL/min)	OPTIONAL	The flow rate. Default the head's own.
<code>direction</code>	string	OPTIONAL	<code>forward</code> or <code>reverse</code> , for a head with the <code>reverse</code> feature. Default <code>forward</code> .
<code>check</code>	boolean	OPTIONAL	<code>true</code> to check the dose without giving it. Default <code>false</code> .

Result	Type	Presence	Description
<code>warnings</code>	array	REQUIRED	What the controller warns the person of, such as a dose larger than the head's largest planned dose (Dosing Section 10.6.4).

actions.prime: request, from an app or service. Primes a dosing head, with its `prime` command (Dosing Section 10.6). Operation: `dose-now`. Audit record: `dose-now`, with the device and the endpoint as `target.device` and `target.endpoint`, and `{"prime": duration}` in `details`. Reasons it may give (Table 2): `bad-endpoint` with `unknown-device`; `bad-command` with `wrong-class`; `bad-state` with `held`, `device-lost`, `device-refused`, `device-unanswered`, and `limit-reached`; `local-control` with `under-local-control`; `bad-parameter` with `too-long`.

Parameter	Type	Presence	Description
<code>device</code>	string	REQUIRED	The device, as Automation Section 8.5 identifies it.
<code>endpoint</code>	string	REQUIRED	The endpoint.
<code>duration</code>	number (s)	REQUIRED	How long the head runs.

Result: `{}`.

- Each action sends the endpoint the command that Table 13 gives, as Section 7.11 says, when a person asks for it. A controller refuses an endpoint whose class does not have that command with `bad-command` and the reason `wrong-class`.
- A controller MUST refuse an action, with `bad-state` and the reason `held`, while the resolved state of the output bound to the endpoint is `stop` [OAP-AU Section 8.3.1], and while any other hold that Table 13 names holds it. The device does not always refuse such a command itself, and a command that starts an operation would end a latched stop [OAP-DC Section 7.4.13].
- The controller counts and records what an action does as the equipment's own specification says, as Table 13 shows.

Table 13 — Actions

Method	Endpoint	Command	Operation	Also held by	Counted and recorded
actions.feed	feeder	feed, with portions	run-action	—	—
actions.test	analyzer	test, with parameters and standard	run-action	A hold of the analyzer, and what keeps a test from starting, except a suspension [OAP-WT Sections 7.6, 8.5, and 8.8.5]	As a test on demand, or a test of a standard [OAP-WT Sections 8.8 and 15.2]
actions.fill	auto-top-off	fill, with duration	run-action, or dose-now for a unit whose water carries an additive [OAP-WL Section 16.2]	A water change, an aborted water change, the leak lockout, and the limits that the controller enforces [OAP-WL Sections 8.13 and 8.14.9]	As every fill [OAP-WL Sections 8.4 and 9.15]
actions.dose	doser	dose, with volume, rate, and direction	dose-now	A latched stuck-on, and an empty container [OAP-DO Sections 11.3 and 14.5.2]	As a manual dose [OAP-DO Sections 10.6.2 and 16.1]
actions.prime	doser	prime, with duration	dose-now	As for actions.dose	Out of the container, not the day's doses [OAP-DO Section 11.1]

- A controller MUST refuse a test that Water Testing Section 8.5 would not start, other than for a suspension, with the reason `held`, and while its clock is not valid with the reason `clock-invalid`. For a test on demand of a suspended analyzer, it enables the analyzer for the test, and holds it again when the test ends, unless an accepted result has ended the suspension [OAP-WT Section 8.8.5].
- A controller MUST refuse a fill while it holds or disables the top-off unit, with the reason `held`, and while the limits that it enforces leave no room, with `bad-state` and the reason `limit-reached` [OAP-WL Sections 8.14.9 and 8.14.10].
- A controller MUST give a dose longer than the head's `maxContinuousRun` in parts, with rests between them [OAP-DO Section 14.7.2], and MUST refuse a prime longer than `maxContinuousRun`, with `bad-parameter`, the reason `too-long`, and the longest as `limit`, and one longer than the head's priming time left in the day, with the reason `limit-reached` [OAP-DO Section 14.7.4; OAP-DC Section 7.14.4].
- A check of `actions.dose` returns the warnings of Dosing Section 10.6.4, such as a dose within the spacing of another head, or one larger than the head's largest planned dose, which the app shows the person before it sends the dose.

16 Calibrations

16.1 Kinds of Calibration

`calibration.start`, `calibration.complete`, and `calibration.cancel` serve each kind of calibration, as Table 14 shows.

Table 14 — Calibrations

Kind	Endpoint	<code>calibration.start</code> sends	<code>calibration.point</code> sends	<code>calibration.complete</code> sends	<code>calibration.cancel</code> sends
Probe	sensor with the calibration feature	<code>beginCalibration</code> , once its inputs are held	<code>calibratePoint</code>	<code>completeCalibration</code>	<code>cancelCalibration</code>
Dosing head	doser	<code>calibrate</code> , with duration and rate	—	<code>completeCalibration</code> , with <code>measuredVolume</code>	<code>stop</code>
Analyzer	analyzer	<code>calibrate</code>	—	— (it ends by itself)	<code>abort</code>

calibration.start: request, from an app or service. Starts the calibration of a probe, a dosing head, or an analyzer (Table 14). Operation: `calibrate`. Audit record: `calibrate`, with the device and the endpoint as `target.device` and `target.endpoint`, and `{"start": true}`, with `duration` and `rate` for a dosing head, in `details`. Reasons it may give (Table 2): `bad-endpoint` with `unknown-device`; `bad-command` with `wrong-class`; `bad-state` with `device-lost`, `device-refused`, `device-unanswered`, `clock-invalid`, and `held`; `bad-parameter` with `unknown-element` and `too-long`; `local-control` with `under-local-control`.

Parameter	Type	Presence	Description
<code>device</code>	string	REQUIRED	The device, as Automation Section 8.5 identifies it.
<code>endpoint</code>	string	REQUIRED	The endpoint.
<code>temperatureInput</code>	identifier	OPTIONAL	For a probe: the Automation input of the temperature probe that the person puts in the references with it (Probe Interface Section 14.7.8).
<code>duration</code>	number (s)	For a dosing head	How long the head runs, within its <code>maxContinuousRun</code> .
<code>rate</code>	number (mL/min)	OPTIONAL	For a dosing head with the <code>calibration-points</code> feature: the flow rate to calibrate.

Result	Type	Presence	Description
<code>left</code>	number (s)	OPTIONAL	For a dosing head, how long its run lasts.

calibration.complete: request, from an app or service. Completes the calibration of a probe or of a dosing head. Operation: `calibrate`. Audit record: `calibrate`, with the device and the endpoint as `target.device` and `target.endpoint`, and the `outcome`, with the flow rates of a dosing head, in `details`. Reasons it may give (Table 2): `bad-endpoint` with `unknown-device`; `bad-command` with `wrong-class`; `bad-state` with `device-lost`, `device-refused`, `device-unanswered`, and `no-calibration`.

Parameter	Type	Presence	Description
<code>device</code>	string	REQUIRED	The device, as Automation Section 8.5 identifies it.
<code>endpoint</code>	string	REQUIRED	The endpoint.
<code>measuredVolume</code>	number (mL)	For a dosing head	The volume that the person measured from the head's run.
<code>confirmed</code>	boolean	OPTIONAL	For a dosing head: <code>true</code> when the person has confirmed a change of its flow rate of more than 10 % (Dosing Section 8.2.5). Default <code>false</code> .

Result	Type	Presence	Description
<code>outcome</code>	string	REQUIRED	<code>completed</code> , <code>failed</code> , or, for a dosing head, <code>needs-confirmation</code> , when the change is larger than 10 % and not <code>confirmed</code> .
<code>record</code>	object	OPTIONAL	The Calibration Record of the calibration (Data Model Section 8.12).
<code>flowRate</code>	number (mL/min)	OPTIONAL	For a dosing head, its flow rate from the run.
<code>previous</code>	number (mL/min)	OPTIONAL	For a dosing head, the flow rate it replaces.

calibration.cancel: request, from an app or service. Cancels a calibration in progress, which keeps the calibration in use. Operation: `calibrate`. Audit record: `calibrate`, with the device and the endpoint as `target.device` and `target.endpoint`, and `{"cancel": true}` in `details`. Reasons it may give (Table 2): `bad-endpoint` with `unknown-device`; `bad-command` with `wrong-class`; `bad-state` with `device-lost`, `device-refused`, and `device-unanswered`.

Parameter	Type	Presence	Description
<code>device</code>	string	REQUIRED	The device, as Automation Section 8.5 identifies it.
<code>endpoint</code>	string	REQUIRED	The endpoint.

Result: `{}`.

A controller refuses a request that Table 14 does not give for the endpoint's class with `bad-command` and the reason `wrong-class`, and a calibration request of an endpoint with no calibration in progress, other than `calibration.start` and `calibration.reset`, with `bad-state` and the reason `no-calibration`.

16.2 Following Calibrations

calibration.list: request, from an app or service. Returns every calibration in progress, and every calibration hold that waits for a person. Operation: `view-state`.

Parameters: none.

Result	Type	Presence	Description
<code>calibrations</code>	array	REQUIRED	Each, with the members of Table 15.

calibration.subscribe: request, from an app or service. Starts `calibration.changed` notifications on the session. Operation: `view-state`.

Parameters: none.

Result: `{}`.

calibration.unsubscribe: request, from an app or service. Ends the session's subscription of the `calibration` namespace. Operation: `view-state`.

Parameters: none.

Result: `{}`.

calibration.changed: notification, from the controller. Reports that a calibration, or a calibration hold, changed.

Parameter	Type	Presence	Description
<code>calibrations</code>	array	REQUIRED	Every calibration and hold, as <code>calibration.list</code> returns them.

1. `calibration.list` MUST return every calibration in progress, and every calibration hold that waits for a person, those that the controller did not start among them, such as one started at a probe's own controls or found at a restart [OAP-PI Sections 14.2.12 and 14.7.6], with the members of Table 15.
2. After it responds to `calibration.subscribe`, a controller MUST send `calibration.changed` within 1 s of each change of what `calibration.list` would return, other than `value`, `left`, and `stable`, but not more than once a second.

Table 15 — Members of a calibration's status

Member	Type	Presence	Description
<code>device, endpoint</code>	string	REQUIRED	The endpoint.
<code>kind</code>	string	REQUIRED	<code>probe</code> , <code>head</code> , or <code>analyzer</code> (Table 14).
<code>phase</code>	string	REQUIRED	For a probe: <code>calibrating</code> , from its start to its end; <code>waiting-for-probe</code> , until a person confirms that the probe is back; <code>settling</code> , for the settle time after it [OAP-PI Section 14.8.1]; <code>new-reading</code> , until a person accepts the reading after a shift [OAP-PI Section 14.8.3]; <code>new-probe</code> , until a person accepts a smart probe that replaced another [OAP-PI Section 16.4.2]; or <code>held</code> , until a calibration is accepted [OAP-PI Section 16.5.2]. For a dosing head: <code>running</code> , while it runs, and <code>measuring</code> , until the person gives the volume. For an analyzer: <code>calibrating</code> .
<code>started</code>	timestamp or null	OPTIONAL	When it started.
<code>points</code>	array	OPTIONAL	For a probe, each point recorded: its <code>reference</code> , and, where known, its <code>reading</code> and <code>temperature</code> .
<code>value, stable</code>	number, boolean	OPTIONAL	For a probe, its reading now, and whether it is stable [OAP-PI Section 14.3].
<code>temperature</code>	number (°C)	OPTIONAL	The reference's temperature, from the temperature probe in it.
<code>before, after</code>	number	OPTIONAL	In <code>new-reading</code> , the readings before and after the calibration.
<code>left</code>	number (s)	OPTIONAL	For a dosing head, how long its run has left.
<code>inputs</code>	array	OPTIONAL	The Automation inputs that the calibration holds.
<code>extensions</code>	object	OPTIONAL	Extensions.

16.3 Probes

calibration.point: request, from an app or service. Records one point of a probe's calibration (Probe Interface Section 14.7.2). Operation: `calibrate`. Audit record: `calibrate`, with the device and the endpoint as `target.device` and `target.endpoint`, and `reference` and `solution` in details; a check is not recorded. Reasons it may give (Table 2): `bad-endpoint` with `unknown-device`; `bad-command` with `wrong-class`; `bad-state` with `device-lost`, `device-refused`, `device-unanswered`, and `no-calibration`; `bad-parameter` with `reference-needed`.

Parameter	Type	Presence	Description
<code>device</code>	string	REQUIRED	The device, as Automation Section 8.5 identifies it.
<code>endpoint</code>	string	REQUIRED	The endpoint.
<code>reference</code>	number	OPTIONAL	The reference's value, in the endpoint's unit. Without it, the controller works it out from <code>solution</code> , <code>temperature</code> , and <code>pressure</code> .
<code>solution</code>	string	OPTIONAL	The reference solution, as Probe Interface Table 20 names it, or an extension name.
<code>temperature</code>	number (°C)	OPTIONAL	The reference's temperature, when no temperature probe is in it.
<code>pressure</code>	number (kPa)	OPTIONAL	For dissolved oxygen, the barometric pressure at the probe (Probe Interface Section 15.5).
<code>check</code>	boolean	OPTIONAL	<code>true</code> to work out the reference's value, and whether readings are stable, without recording the point. Default <code>false</code> .

Result	Type	Presence	Description
<code>reference</code>	number	REQUIRED	The reference's value, as recorded, or as the controller proposes it.
<code>stable</code>	boolean	REQUIRED	Whether the probe's readings are stable (Probe Interface Section 14.3).

calibration.confirm: request, from an app or service. Gives a person's confirmation that a calibration hold waits for (Probe Interface Section 14.8). Operation: `calibrate`. Audit record: `calibrate`, with the device and the endpoint as `target.device` and `target.endpoint`, and `confirm` in details. Reasons it may give (Table 2): `bad-endpoint` with `unknown-device`; `bad-command` with `wrong-class`; `bad-state` with `device-lost`, `device-refused`, `device-unanswered`, and `no-calibration`.

Parameter	Type	Presence	Description
<code>device</code>	string	REQUIRED	The device, as Automation Section 8.5 identifies it.
<code>endpoint</code>	string	REQUIRED	The endpoint.
<code>confirm</code>	string	REQUIRED	<code>probe-back</code> : the probe, and any temperature probe used, is back in the system; <code>new-reading</code> : the person accepts the reading after a shift; or <code>new-probe</code> : the person accepts a smart probe that replaced another.

Result: `{}`.

calibration.reset: request, from an app or service. Resets a probe's calibration, for a probe that a person replaced (Probe Interface Section 16.5.2). Operation: `calibrate`. Audit record: `calibrate`, with the device and the endpoint as `target.device` and `target.endpoint`, and `{"reset": true}` in details. Reasons it may give (Table 2): `bad-endpoint` with `unknown-device`; `bad-command` with `wrong-class`; `bad-state` with `device-lost`, `device-refused`, and `device-unanswered`.

Parameter	Type	Presence	Description
<code>device</code>	string	REQUIRED	The device, as Automation Section 8.5 identifies it.
<code>endpoint</code>	string	REQUIRED	The endpoint.

Result: `{}`.

A probe's calibration follows Probe Interface Section 14. The app guides the person through it: which functions use the probe's inputs, and what they do while the inputs are unavailable [OAP-PI Section 14.7.3]; each reference in turn; and the result.

1. For a probe, `calibration.start` MUST hold the inputs of every endpoint of the probe's port, and of the temperature probe of `temperatureInput`, from when it starts, before it sends `beginCalibration` [OAP-PI

- Sections 14.7.4 and 14.7.8], and a controller refuses it with `bad-state` and the reason `clock-invalid` while its clock is not valid [OAP-PI Section 14.2.1].
2. While a probe is calibrated, a controller that writes the port's compensation temperature MUST write the reference's, not the system's, and MUST write the system's again when the calibration ends [OAP-PI Section 14.4.5].
 3. With `check true`, `calibration.point` works out the reference's value at its temperature from `solution`, `temperature`, and `pressure`, as Probe Interface Sections 14.4 and 15 say, for the person to confirm [OAP-PI Section 14.7.2], and says whether the readings are stable; without `check`, it records the point with that value, or with `reference`. A controller MUST convert an ORP reference given against another reference electrode before it records it [OAP-PI Section 15.2.2], and refuses a point whose value it cannot work out, with `bad-parameter` and the reason `reference-needed`.
 4. An app SHOULD record a point only once the readings are stable [OAP-PI Section 14.3.4].
 5. `calibration.complete` returns the outcome, and the Calibration Record that the port reports [OAP-PI Section 14.9.1].
 6. A controller MUST NOT end a probe's calibration hold until the calibration has ended, a person has confirmed with `calibration.confirm` that the probe is back, and the readings have been good and stable for the settle time, and, after a shift larger than the probe type's, until a person has accepted the new reading [OAP-PI Sections 14.8.1 and 14.8.3]. A controller SHOULD NOT accept `new-reading` in a session through a relay, from a person who is not at the system, and SHOULD say so when it does [OAP-PI Section 20.3].
 7. `calibration.reset` sends `resetCalibration`, for a probe without memory that a person replaced, and the controller holds the probe's inputs, as `held`, when the port cannot reset, until a calibration is accepted [OAP-PI Section 16.5.2].

The controller records each calibration, and sets the probe's next due date, as Probe Interface Section 14.9 requires.

16.4 Dosing Heads

For a dosing head, `calibration.start`, `calibration.complete`, and `calibration.cancel` carry out the calibration of Dosing Section 8.2.

1. A controller MUST refuse `calibration.start` of a head while it is held as Section 15.1.2 says, with `bad-state` and the reason `held` [OAP-DO Section 10.6.3], and with a `duration` longer than the head's `maxContinuousRun`, with `bad-parameter`, the reason `too-long`, and the longest as `limit` [OAP-DO Section 14.7.4].
2. An app SHOULD propose a `duration` that delivers at least 10 mL [OAP-DO Section 8.2.4].
3. With `measuredVolume`, `calibration.complete` works out the head's flow rate from the run, and, when it differs from the previous one by more than 10 % and `confirmed` is not `true`, MUST send nothing, and returns the outcome `needs-confirmation`, with both rates, for the person to confirm [OAP-DO Section 8.2.5]; otherwise it sends `completeCalibration`.
4. `calibration.cancel` stops the run, which does not resume [OAP-DC Section 7.14.1].

The controller records each calibration of a head as Dosing Section 16.3.1 says.

16.5 Analyzers

1. For an analyzer, `calibration.start` sends its own `calibrate` command, only when a person asks for it, and a controller MUST refuse it while the analyzer is held, with `bad-state` and the reason `held` [OAP-WT Section 7.6].
2. The calibration ends by itself; the analyzer's next result is a fresh start, and its corrections stay as they are [OAP-WT Section 11.8].

16.6 Corrections of Analyzers' Results

`calibration.listCorrections`: request, from an app or service. Returns the corrections of analyzers' results, their comparisons with references, and the corrections the controller proposes (Water Testing Section 11).
Operation: `view-configuration`.

Parameters: none.

Result	Type	Presence	Description
corrections	array	REQUIRED	Each analyzer's parameter that has a correction, a comparison, or a proposal, with the members of Table 16.

calibration.setReference: request, from an app or service. Chooses whether a result is a reference with which analyzers' results are compared (Water Testing Section 11.2). Operation: `calibrate`. Audit record: `calibrate`, with the observation and reference in details. Reasons it may give (Table 2): `bad-parameter` with `unknown-element` and `invalid-reference`.

Parameter	Type	Presence	Description
observation	identifier	REQUIRED	The result, by the <code>id</code> of its observation.
reference	boolean	REQUIRED	<code>true</code> to make it a reference, <code>false</code> to make it none.

Result: `{}`.

calibration.applyCorrection: request, from an app or service. Applies a correction to an analyzer's results, as the controller proposes it or as the person gives it (Water Testing Section 11.5). Operation: `calibrate`. Audit record: `calibrate`, with the function as `target.element`, and the parameter and the correction in details. Reasons it may give (Table 2): `bad-parameter` with `unknown-element`; `bad-state` with `no-proposal`.

Parameter	Type	Presence	Description
function	identifier	REQUIRED	The water testing function.
parameter	string	REQUIRED	The parameter.
method	string	OPTIONAL	<code>factor</code> or <code>offset</code> . Default the proposal's.
factor	number	OPTIONAL	The person's own factor.
offset	number	OPTIONAL	The person's own offset, in unit.
unit	string	With <code>offset</code>	The offset's unit.
comparisons	array	OPTIONAL	The comparisons to use, by the <code>id</code> of their events. Default the proposal's.
confirmed	boolean	OPTIONAL	<code>true</code> when the person has confirmed a correction that changes the latest result by more than <code>maxCorrection</code> , or by more than 10 % without one (Water Testing Section 11.6). Default <code>false</code> .

Result	Type	Presence	Description
status	string	REQUIRED	<code>applied</code> , or <code>needs-confirmation</code> .
correction	object	REQUIRED	The correction, as Water Testing Table 15 defines it, applied or proposed.
change	number (%)	OPTIONAL	How much it changes the latest result.
version	string	OPTIONAL	The configuration document's new version, once applied.

calibration.dismissCorrection: request, from an app or service. Dismisses the correction that the controller proposes for an analyzer's parameter (Water Testing Section 11.4). Operation: `calibrate`. Audit record: `calibrate`, with the function as `target.element`, and the parameter and `{"dismissed": true}` in details. Reasons it may give (Table 2): `bad-parameter` with `unknown-element`; `bad-state` with `no-proposal`.

Parameter	Type	Presence	Description
function	identifier	REQUIRED	The water testing function.
parameter	string	REQUIRED	The parameter.

Result: `{}`.

1. `calibration.listCorrections` MUST return, for each analyzer's parameter, its correction, its comparisons with references [OAP-WT Section 11.3], and the correction that the controller proposes [OAP-WT Section 11.4], with the members of Table 16.
2. A person chooses which results are references, with `calibration.setReference` [OAP-WT Section 11.2]; a controller refuses a result that cannot be one, such as a result that is not confirmed, with `bad-parameter` and the reason `invalid-reference` [OAP-WT Section 12.7].

3. `calibration.applyCorrection` is the person's confirmation of a correction [OAP-WT Section 11.5.1]. Before it sends one, an app **MUST** show the person the comparisons that it comes from, the previous correction, and how much it changes the latest result [OAP-WT Section 11.5.2].
4. When the correction changes the latest result by more than `maxCorrection`, or, without one, by more than 10 %, and `confirmed` is not `true`, a controller **MUST NOT** apply it, and returns the status `needs-confirmation`, with the change, for the person to confirm [OAP-WT Section 11.6.5].
5. A correction is part of the configuration [OAP-WT Section 11.5.8]: applying one gives the configuration document a new version.
6. `calibration.dismissCorrection` dismisses a proposal [OAP-WT Section 11.4.6]; with no proposal, and with no correction of its own in `calibration.applyCorrection`, a controller refuses either with `bad-state` and the reason `no-proposal`.

Table 16 — Members of a correction's status

Member	Type	Presence	Description
<code>function, parameter</code>	identifier, string	REQUIRED	The water testing function, and the parameter of its analyzer.
<code>correction</code>	object	OPTIONAL	The correction in use, as Water Testing Table 15 defines it.
<code>comparisons</code>	array	REQUIRED	The comparisons within <code>compareDays</code> , newest first: each its event's <code>id</code> , its <code>time</code> , the <code>reference</code> , the <code>raw</code> result, the <code>difference</code> , and the <code>ratio</code> [OAP-WT Section 11.3].
<code>proposal</code>	object	OPTIONAL	The proposed correction: its <code>method</code> , <code>factor</code> or <code>offset</code> and <code>unit</code> , the <code>comparisons</code> it comes from, their <code>spread</code> , and the <code>change</code> , in percent, of the latest result [OAP-WT Section 11.4].
<code>overLimit</code>	boolean	OPTIONAL	<code>true</code> while <code>correction-limit</code> is raised [OAP-WT Section 11.6].
<code>extensions</code>	object	OPTIONAL	Extensions.

17 Firmware Updates

17.1 Firmware and Releases

`updates.list`: request, from an app or service. Returns the firmware of the controller and of each of its devices and modules, and the updates available for each (Security and Access Section 15.9.2). Operation: `view-configuration`.

Parameters: none.

Result	Type	Presence	Description
<code>units</code>	array	REQUIRED	The controller, and each device and module, with the members of Table 17.

1. `updates.list` **MUST** return the controller and each of its devices and modules, with what Security and Access Section 15.9.2 requires a controller to show, with the members of Table 17, and the releases available to each: from the feeds that it checks with an owner's consent [OAP-SA Section 16.2], and from the manifests that apps have given it.
2. A controller checks a feed only after an owner has consented to update checks, and sends it nothing that identifies the unit [OAP-SA Table 12].

Table 17 — Members of a unit's firmware

Member	Type	Presence	Description
<code>device</code>	string	OPTIONAL	The device. Absent for the controller.
<code>manufacturer, model, hardwareRevision</code>	string	<code>hardwareRevision</code> OPTIONAL	The unit.
<code>firmwareVersion, securityVersion</code>	string, integer	<code>securityVersion</code> OPTIONAL	Its firmware [OAP-SA Section 15.9].
<code>channel, signer</code>	string	OPTIONAL	Its update channel, and who signed its firmware: <code>manufacturer</code> or <code>owner</code> [OAP-SA Section 15.8.3].

Member	Type	Presence	Description
<code>automaticInstall</code> , <code>window</code> , <code>postponeUntil</code>	boolean, object, timestamp	OPTIONAL	Its automatic installation (Section 17.3).
<code>supportEnds</code>	date	OPTIONAL	The end of its security support.
<code>lastUpdate</code>	object	OPTIONAL	Its last update, as Device Classes Section 8.1 defines <code>LastUpdate</code> .
<code>available</code>	array	REQUIRED	Each release newer than its firmware that its security version allows, as its signed <code>manifest</code> and the manifest's payload, as <code>release</code> .
<code>checked</code>	timestamp	OPTIONAL	When the controller last checked its feed.
<code>extensions</code>	object	OPTIONAL	Extensions.

17.2 Installing

updates.upload: request, from an app or service. Sends the controller part of a firmware image to install, for an update without the internet (Security and Access Section 15.5.4). Operation: `install-firmware`. Audit record: none of its own; `updates.install` records the installation. Reasons it may give (Table 2): `bad-parameter` with `unknown-file`, `file-too-large`, and `out-of-order`.

Parameter	Type	Presence	Description
<code>file</code>	string	After the first part	The file's name, from the result of its first part.
<code>offset</code>	integer	REQUIRED	Where the part goes, in bytes: the number of bytes received so far.
<code>data</code>	string	REQUIRED	The bytes, in base64url.
<code>size</code>	integer	In the first part	The size of the whole image, in bytes.
<code>sha256</code>	string	OPTIONAL	In the first part, the whole image's SHA-256 hash, in lowercase hexadecimal.

Result	Type	Presence	Description
<code>file</code>	string	REQUIRED	The file's name.
<code>received</code>	integer	REQUIRED	The number of bytes received so far.

updates.install: request, from an app or service. Checks an update of the controller or of a device, or starts installing it. Operation: `install-firmware`. Audit record: `install-firmware`, with the device as `target.device`, or no target for the controller, and the manifest's `version` and the version it replaces in `details`, as `to` and `from`; a check is not recorded. Reasons it may give (Table 2): `bad-endpoint` with `unknown-device`; `bad-parameter` with `unknown-file` and `manifest-refused`; `bad-state` with `incomplete`, `operation-running`, and `device-lost`; `busy` with `update-running`.

Parameter	Type	Presence	Description
<code>device</code>	string	OPTIONAL	The device. Without it, the controller.
<code>manifest</code>	string	REQUIRED	The release's signed update manifest (Security and Access Section 15.6), from <code>updates.list</code> , or from the person.
<code>file</code>	string	OPTIONAL	An image that the session uploaded. Without it, the controller downloads the image from the manifest's <code>image.url</code> .
<code>check</code>	boolean	OPTIONAL	<code>true</code> to check the update without starting it. Default <code>false</code> .

Result	Type	Presence	Description
<code>operation</code>	string	OPTIONAL	The operation's identifier, which <code>updates.progress</code> reports (Section 7.12). Absent for a check.
<code>release</code>	object	REQUIRED	The manifest's payload (Security and Access Table 10).
<code>warnings</code>	array	REQUIRED	What the controller warns the person of, such as an interruption longer than the fail-safe timeouts of its devices (Fail-Safe Behavior Section 11.2.4).

updates.progress: notification, from the controller. Reports the progress of an operation of `updates.install`, to the session that started it.

Parameter	Type	Presence	Description
operation	string	REQUIRED	The operation.
device	string	OPTIONAL	The device. Absent for the controller.
state	string	REQUIRED	waiting (for an operation of the device to end), downloading, verifying, installing, restarting, or, at the end, installed, refused, reverted, or failed.
received	integer	OPTIONAL	While downloading or sending the image, the bytes so far.
size	integer	OPTIONAL	The image's size, in bytes.
reason	string	OPTIONAL	Why it was refused, reverted, or failed, for people.

1. Before it sends a request without `check`, which is the user's approval of the update [OAP-SA Section 15.5.1], an app MUST show the person what Security and Access Section 15.5.3 requires, from the result of a check: the version, the channel, the release notes, the security fixes, how long the unit will not be under control, and what its outputs do meanwhile, with the warnings.
2. A controller MUST check the manifest and the image as Security and Access Section 15.6.6 requires before it installs the image or sends it to a device, and MUST refuse an update that fails, with `bad-parameter`, the reason `manifest-refused`, and why in `errors`, and record `firmware-refused`.
3. A controller MUST refuse an update of a device while one of its endpoints is dosing, filling, or taking part in a water change, and of itself while an endpoint that it controls is, with `bad-state` and the reason `operation-running` [OAP-SA Section 15.5.2; OAP-FS Section 11.2.2].
4. The image is the file that the session uploaded with `updates.upload`, as `export.upload` uploads a file (Section 14.2.1), or, without `file`, the controller downloads it from the manifest's `image.url` once the user has approved the update [OAP-SA Table 12], so that an owner can update without the internet [OAP-SA Section 15.5.4].
5. A controller MUST send `updates.progress` whenever the operation's `state` changes, and at least every 5 s while it downloads or sends the image, as a device reports to it [OAP-DP Section 13.1.3; OAP-MB Section 13.7].
6. For its own update, a controller MUST keep its control functions running while it writes the image, and restart only once the image is written and checked, and then confirms the new firmware as Security and Access Section 15.4.2 says, or returns to the previous one. A check MUST warn of an update whose `interruption` is longer than the fail-safe timeouts of its devices [OAP-FS Section 11.2.4].

17.3 Settings and Reverting

updates.configure: request, from an app or service. Changes the update channel and the automatic installation of the controller or of a device, or postpones an update (Security and Access Sections 15.5 and 15.7). Operation: `install-firmware`. Audit record: `install-firmware`, with the device as `target.device`, or no target for the controller, and each change in `details`, as `changes`. Reasons it may give (Table 2): `bad-endpoint` with `unknown-device`; `bad-state` with `no-consent`, `device-lost`, `device-refused`, and `device-unanswered`.

Parameter	Type	Presence	Description
device	string	OPTIONAL	The device. Without it, the controller.
channel	string	OPTIONAL	<code>stable</code> or <code>beta</code> .
automaticInstall	boolean	OPTIONAL	Whether updates of the channel install automatically.
window	object	OPTIONAL	The local times of day within which automatic installation may start, as <code>start</code> and <code>end</code> , or <code>null</code> for none.
postponeUntil	timestamp or null	OPTIONAL	No update installs automatically before this time; <code>null</code> ends a postponement.

Result	Type	Presence	Description
channel	string	REQUIRED	The channel now.
automaticInstall	boolean	REQUIRED	Whether updates install automatically now.
window	object	OPTIONAL	The window now.
postponeUntil	timestamp	OPTIONAL	The postponement now.

updates.revert: request, from an app or service. Returns the controller or a device to its previous firmware (Security and Access Section 15.4.4). Operation: `install-firmware`. Audit record: `install-firmware`, with the device as `target.device`, or no target for the controller, and `{"revert": true}` in `details`. Reasons it may give (Table 2): `bad-endpoint` with `unknown-device`; `bad-state` with `no-previous`, `operation-running`, `device-lost`, `device-refused`, and `device-unanswered`; `busy` with `update-running`.

Parameter	Type	Presence	Description
<code>device</code>	string	OPTIONAL	The device. Without it, the controller.

Result: `{}`.

1. `updates.configure` sets the update channel, which stays `stable` until a user chooses `beta` [OAP-SA Section 15.7.2], and automatic installation, which stays off until a user turns it on, within `window`, where there is one, and not before `postponeUntil` [OAP-SA Sections 15.5.1 and 15.5.2]. For a device, the controller writes its `updateChannel` and `automaticInstall` [OAP-DP Section 13.1.8].
2. A controller MUST refuse to turn automatic installation on before an owner has consented to update checks, with `bad-state` and the reason `no-consent` [OAP-SA Section 15.5.1].
3. `updates.revert` returns the unit to its previous firmware [OAP-SA Section 15.4.4], and a controller MUST refuse it, with `bad-state`, when the unit has no previous firmware that its security version allows, with the reason `no-previous`, and while an endpoint of the unit is dosing, filling, or taking part in a water change, with the reason `operation-running` [OAP-DP Section 13.1.7].

18 Remote and Emergency Access

18.1 Remote Access

access.readRemote: request, from an app or service. Returns whether remote access is on, and its relays (Security and Access Section 13). Operation: `view-configuration`.

Parameters: none.

Result	Type	Presence	Description
<code>enabled</code>	boolean	REQUIRED	Whether remote access is on.
<code>relays</code>	array	REQUIRED	The relays the controller listens on, each with its <code>url</code> and, where the relay is checked by key, its <code>keyId</code> , and <code>connected</code> , whether the controller's session with it is open now (Device Protocol Section 15.4).

access.configureRemote: request, from an app or service. Turns remote access on or off, and sets its relays. Operation: `configure-remote-access`. Audit record: `configure-remote-access`, with `enabled` and the relays' URLs in `details`.

Parameter	Type	Presence	Description
<code>enabled</code>	boolean	OPTIONAL	Whether remote access is on.
<code>relays</code>	array	OPTIONAL	The relays, each with a <code>url</code> and, optionally, a <code>keyId</code> .

Result	Type	Presence	Description
<code>enabled</code>	boolean	REQUIRED	Whether remote access is on now.
<code>relays</code>	array	REQUIRED	The relays now.

1. Remote access is off until an owner turns it on [OAP-SA Section 13.1.2]. When an owner turns it off, a controller MUST stop it within 5 s: it closes its relay sessions, and every remote session [OAP-SA Section 13.1.2].
2. A controller listens on each relay as Device Protocol Section 15.4.3 says, and tells apps whose users may connect remotely which relays it listens on, in the result of `session.hello` [OAP-DP Table 6].

18.2 Emergency Access

access.readEmergency: request, from an app or service. Returns whether emergency access is on and available, how to join its link, and its emergency codes (Security and Access Section 14). Operation: `view-configuration`.

Parameters: none.

Result	Type	Presence	Description
<code>enabled</code>	boolean	REQUIRED	Whether emergency access is on.
<code>available</code>	boolean	REQUIRED	Whether its link is available now (Security and Access Section 14.3.2).
<code>locked</code>	boolean	REQUIRED	Whether emergency codes are refused after failed attempts (Security and Access Section 14.4.3).
<code>ssid</code>	string	OPTIONAL	The name of the emergency link's network.
<code>passphrase</code>	string	OPTIONAL	Its passphrase, only for a user whose role is <code>override</code> or above, outside an emergency session (Section 18.2).
<code>codes</code>	array	REQUIRED	Each emergency code, without the code: its <code>slot</code> , its <code>user</code> , the user's <code>name</code> , and when it was created.

access.configureEmergency: request, from an app or service. Turns emergency access on or off, or allows emergency codes again after failed attempts. Operation: `configure-emergency-access`. Audit record: `configure-emergency-access`, with `enabled` or `unlock` in details.

Parameter	Type	Presence	Description
<code>enabled</code>	boolean	OPTIONAL	Whether emergency access is on.
<code>unlock</code>	boolean	OPTIONAL	<code>true</code> to allow emergency codes again after failed attempts.

Result	Type	Presence	Description
<code>enabled</code>	boolean	REQUIRED	Whether emergency access is on now.
<code>locked</code>	boolean	REQUIRED	Whether emergency codes are refused now.

access.createEmergencyCode: request, from an app or service. Creates an emergency code for a person without an app, such as a neighbor, as a new user with the `override` role (Security and Access Section 14.4.2). Operation: `configure-emergency-access`. Audit record: `configure-emergency-access`, with the new user as `target.user`, and the `slot` in details; never the code.

Parameter	Type	Presence	Description
<code>name</code>	string	REQUIRED	The person's name.

Result	Type	Presence	Description
<code>user</code>	UUID	REQUIRED	The new user's <code>id</code> .
<code>slot</code>	integer	REQUIRED	The code's <code>slot</code> , from 0 to 99.
<code>code</code>	string	REQUIRED	The digits that follow the <code>slot</code> , at least eight, shown only this once.

access.removeEmergencyCode: request, from an app or service. Removes an emergency code, and its user. Operation: `configure-emergency-access`. Audit record: `configure-emergency-access`, with the code's `user` as `target.user`, and the `slot` and `{"removed": true}` in details. Reasons it may give (Table 2): `bad-parameter` with `unknown-slot`.

Parameter	Type	Presence	Description
<code>slot</code>	integer	REQUIRED	The code's <code>slot</code> .

Result: `{}`.

- Emergency access is off until an owner turns it on [OAP-SA Section 14.3.1].
- A controller MUST give the passphrase of its emergency link in `access.readEmergency` only to a user whose role is `override` or above, outside an emergency session, so that paired apps receive it over a secure channel [OAP-SA Section 14.6.1], and to no one else.
- `access.createEmergencyCode` creates a user of kind `person`, with the `override` role, whose only credential is an emergency code that works only for emergency access [OAP-SA Section 14.4.2]. A controller MUST

make the code of at least eight random digits, with a slot that no other emergency code has [OAP-DP Section 10.3.2], MUST return the code only in that result, and MUST keep only its verifier [OAP-SA Section 9.2.4].

4. `access.configureEmergency` with `unlock` allows emergency codes again after failed attempts [OAP-SA Section 14.4.3].
5. `access.removeEmergencyCode` removes the code and its user, and a controller MUST close an emergency session that the code opened within 5 s [OAP-SA Section 10.6.1].

19 Privacy

19.1 Consent

privacy.listConsent: request, from an app or service. Returns each purpose for which the controller, or a device, would send data off the owner's network, and whether an owner has consented (Security and Access Section 16.3). Operation: `view-configuration`. Reasons it may give (Table 2): `bad-endpoint` with `unknown-device`; `bad-state` with `device-lost` and `device-unanswered`.

Parameter	Type	Presence	Description
<code>device</code>	string	OPTIONAL	The device. Without it, the controller.

Result	Type	Presence	Description
<code>purposes</code>	array	REQUIRED	Each purpose, with the members of Table 18.

privacy.setConsent: request, from an app or service. Gives or withdraws an owner's consent for one purpose, with effect at once. Operation: `manage-consent`. Audit record: `manage-consent`, with the device as `target.device`, or no target for the controller, and `purpose` and `given` in `details`. Reasons it may give (Table 2): `bad-endpoint` with `unknown-device`; `bad-parameter` with `unknown-purpose`; `bad-state` with `device-lost`, `device-refused`, and `device-unanswered`.

Parameter	Type	Presence	Description
<code>device</code>	string	OPTIONAL	The device. Without it, the controller.
<code>purpose</code>	string	REQUIRED	The purpose's <code>id</code> .
<code>given</code>	boolean	REQUIRED	<code>true</code> to consent, <code>false</code> to withdraw or decline.

Result: `{}`.

1. `privacy.listConsent` MUST return each purpose for which the controller would send data off the owner's network, with what Security and Access Section 16.3.2 requires a controller to say before asking, as the members of Table 18, and `given null` for a purpose that no owner has answered yet [OAP-SA Section 16.2.2]. For a device, the controller returns what the device's `consent.list` gives [OAP-DP Section 13.5.2].
2. `privacy.setConsent` gives or withdraws consent with effect at once [OAP-SA Section 16.3.3], and withdrawing it takes away only the function that needs the data [OAP-SA Section 16.3.4]. For a device, the controller sends its `consent.set` [OAP-DP Section 13.5.3].

Table 18 — Members of a purpose

Member	Type	Presence	Description
<code>id</code>	string	REQUIRED	The purpose, such as <code>update-checks</code> .
<code>description</code>	string	REQUIRED	What it is for, for people.
<code>destination</code>	string	REQUIRED	Who receives the data.
<code>data, frequency, retention</code>	string	OPTIONAL	What data is sent, how often, and how long the receiver keeps it [OAP-SA Section 16.3.2].
<code>given</code>	boolean or <code>null</code>	REQUIRED	Whether an owner has consented, or <code>null</code> while no owner has answered.
<code>extensions</code>	object	OPTIONAL	Extensions.

19.2 Erasing Data

privacy.eraseData: request, from an app or service. Erases the controller's history and other personal data, without a factory reset (Security and Access Section 16.6.2). Operation: `erase-data`. Audit record: `erase-data`, which stays in the audit log, which erasing data does not erase.

Parameters: none.

Result: `{}`.

1. `privacy.eraseData` MUST erase the controller's history, its alarm records among it [OAP-AN Section 20.4.2], and the files it holds for sessions, without a factory reset, and MUST keep its audit log, in which it records the erasure [OAP-SA Section 16.6.2].
2. It keeps the configuration, the alarm settings, the users, and their credentials, so that the system goes on running; an owner who wants these erased too factory resets the controller (Section 8.7).

20 Conformance

20.1 Controllers

A conforming controller:

1. meets Sections 7 to 19 as they apply to controllers;
2. serves every request of Table 3 that an app or service sends it, and sends the notifications of Table 3, and `alarms.notify`, as Sections 7 to 19 require;
3. meets the Device Protocol's requirements for controllers [OAP-DP Section 17.2], and Security and Access Section 19.2; and
4. passes the tests of Table 19 that apply to controllers.

There is one conformance level for controllers: a controller conforms when it implements every method and every requirement of this Specification that applies to it, with no reduced profile for small controllers.

20.2 Apps and Services

A conforming app or service:

1. meets the requirements of Section 7 that apply to apps, and those of Sections 8 to 19 for each method that it uses, among them Sections 7.5.2, 7.6.6, 8.5.7, 11.2.2, 16.6.3, and 17.2.1;
2. meets the Device Protocol's requirements for apps and services [OAP-DP Section 17.3];
3. as a Matter bridge on another host, uses `fabric` as Section 10.9 says, and meets Matter Mapping Section 7.1; and
4. passes the tests of Table 19 that apply to apps.

20.3 Machine-Readable Definitions and Tools

`methods.json` is normative. It defines the namespaces, the methods, with the operation each needs, its audit record, its parameters, its result, and the reasons it may give, and the reasons of Table 2. Tables 1 to 3, and the definition of each method in Sections 8 to 19, are generated from it by `tools/render_spec_tables.py`, which also checks it against Security and Access `permissions.json`, the schema, and the Device Protocol's `protocol.json`.

The JSON Schema `schema/controller-api.schema.json` [JSON Schema] is normative for the structure of messages: the parameters of each method, as `<method>.params`, its result, as `<method>.result`, and the data of its errors, as `errorData`. Its `$id` is <https://openaquariumproject.org/schemas/controller-api/0.1>. It refers to the schemas of the Device Protocol, for the envelope of messages, and of the Data Model, Device Classes, Automation, Lighting Programs, Alarms and Notifications, and Security and Access, for the documents that the methods carry. Where the schema and this document differ, this document governs.

The validator, `tools/validate_controller_api.py`, is informative. It checks recorded message exchanges against the schema, and against rules that a schema cannot express: that sessions begin with `session.hello`, that a controller accepts messages of 1 048 576 bytes and no party sends a message larger than the other

accepts, that responses answer outstanding requests and match their methods, that methods come from the parties that send them, that a request succeeds only when the role of the session's user permits its operation, in an emergency session only when the `override` role permits it, and while the clock is not valid only as Section 7.2.4 allows, that errors carry the code that Table 2 gives their reason, that notifications reach only sessions with a subscription of their namespace, and `alarms.notify` only sessions that asked for it, that `fabric` comes only from a Matter bridge's user, and push subscriptions only from sessions that a credential authenticated, that a replacement with `ifVersion` succeeds only when it matches the document's version, and that the documents and records that the methods carry are valid against the validators of their own specifications. The `examples` directory contains message exchanges that pass it (Appendix B).

20.4 Test Procedures

These tests are in addition to those of the Device Protocol and Security and Access specifications, which also apply. Tests use a controller with modules or simulated devices whose readings and faults the tester can set, a clock the tester can set, users of each role, a recording of every session, and a push service under the tester's control. Each test names the requirements it checks. Every MUST and MUST NOT of Sections 7 to 19 is checked by at least one test. A pass criterion that checks what a SHOULD asks applies to a product that does it.

Table 19 — Test procedures

Test	Requirements	Method	Pass criteria
Access	7.1.2, 7.2.1, 7.4.4	From apps of users of each role, on the local network and through a relay, send each request of Table 3; then change a user's role while its session is open, and send them again on that session.	Every request whose operation the user's role permits is served; every other is refused with <code>not-permitted</code> , and changes nothing. After the change, the session's requests are checked against the new role.
Emergency sessions	7.2.3	In an emergency session of an owner, send <code>automation.stopAll</code> , <code>automation.setOverride</code> , <code>alarms.acknowledge</code> , <code>automation.replaceConfiguration</code> , <code>users.update</code> , and <code>alarms.registerPush</code> .	The first three are carried out. The others are refused with <code>not-permitted</code> and the reason <code>emergency-session</code> .
Grants without a clock	7.2.4	Restart the controller with its clock not valid. From a user of the <code>override</code> role whose grant has an expiry, send <code>alarms.list</code> , <code>automation.setOverride</code> with stop and with run, <code>automation.stopAll</code> , <code>automation.endStopAll</code> , <code>alarms.acknowledge</code> , and <code>history.read</code> .	<code>alarms.list</code> , the override with stop, and both requests of <code>stop-all</code> are carried out. The others are refused with <code>not-permitted</code> and the reason <code>grant-unverified</code> .
Audit records	7.3.1, 7.3.2	Send each audited method of Table 3 once so that it succeeds, once from a role that it does not permit, and once with a value that it refuses, on the local network, through a relay, in an emergency session, and from a Matter bridge's user with <code>fabric</code> , with a webhook channel that has a secret, a user with a password, and a push subscription. Read the audit log.	There is one record for each request, with the action, target, and details that its section gives, the session's user as actor, via <code>local</code> , <code>remote</code> , <code>emergency</code> , or <code>matter</code> , and <code>result</code> <code>success</code> , <code>denied</code> , or <code>failed</code> . No record holds a password, a webhook secret, a <code>p256dh</code> , an <code>auth</code> , a code, or a key.
Errors	7.4.2, 7.4.3	Send each method with parameters that break its definition in the schema, and with values that give each reason that its section lists.	The first are refused with <code>invalid-params</code> . Each of the others is refused with the error that Table 2 gives its reason, with <code>data</code> valid against <code>errorData</code> holding the reason, and <code>errors</code> where the reason names them.
Nothing in part	7.4.4, 11.3.2, 14.2.4	Send <code>alarms.acknowledge</code> with the <code>ids</code> of two alarms and one unknown <code>id</code> ; <code>users.update</code> that renames the last owner and gives it the <code>view</code> role; and <code>export.import</code> of a file with a valid configuration document and an alarm settings document that is not valid.	Each is refused, and nothing changes: neither alarm is acknowledged, the owner keeps its name, and neither the configuration nor the alarm settings change.
Message sizes	7.5.1 to 7.5.4	Read the <code>maxMessageSize</code> of the controller's <code>session.hello</code> , and of an app that reads	Both accept at least 1 048 576 bytes. The first document is accepted, and

Test	Requirements	Method	Pass criteria
		configuration documents. Replace the configuration with a document of 999 000 bytes, read it back, and then replace it with one of 1 000 500 bytes. From an app that accepts messages of 131 072 bytes, read history and the audit log that are larger than that.	read back in one message; the second is refused with <code>invalid-document</code> , and a problem that names its size. Every page and every response of the audit log fits in 131 072 bytes, with <code>next</code> or <code>more</code> , and none is refused with <code>too-large</code> .
Versions	7.6.1, 7.6.2	Read the configuration document and the alarm settings. Change each from another app, then by the controller's own means, and then restart the controller without changing them. Replace each with the version first read as <code>ifVersion</code> .	Each version changes with each change, and stays the same across the restart. Both replacements are refused with <code>bad-state</code> and the reason <code>version-conflict</code> , and change nothing.
Checks	7.6.3, 7.6.5, 10.1.2, 11.6.2	Check a configuration document with an element of an extension type that the controller does not support, an output bound to a device it does not have, a heater whose cutoff reads only a sensor on Wi-Fi, and an input whose maximum age is shorter than three reporting intervals of its device. Check alarm settings that send critical alarms to nobody and name an output that the configuration does not have. With <code>check true</code> , check a manual result, a laboratory document, and an export file. Compare the state, history, and audit log before and after.	The configuration has the warnings <code>unsupported</code> , <code>binding-missing</code> , and <code>safety-path-missing</code> , and <code>warning</code> for the maximum age, each naming its element. The alarm settings have a warning for the critical alarms and one for the reference. Nothing changed, and nothing was recorded.
Confirmation in the app	7.6.6	With an app, import a configuration document and alarm settings from files, and import an export file.	Before it sends each request, the app shows the outputs that the configuration drives and how, each binding that it did not make itself, the warnings of the check, and each recipient and channel through which alarms would leave the owner's network, and it sends the request only after the person confirms.
Subscriptions	7.7.2	On one session, subscribe to the <code>system</code> , <code>automation</code> , and <code>alarms</code> namespaces, and then unsubscribe from <code>system</code> . Open a second session without subscriptions. Change a device's state, the saved state, and an alarm in quick succession.	Notifications of each namespace arrive only on the first session, only while it is subscribed to that namespace, after the response to the subscription, and in the order of the changes. None arrive on the second session.
Control while apps are busy	7.7.3, 8.3.2	Open 8 sessions subscribed to every namespace, and have them read history, the configuration, and the state of devices as fast as they can. Record the controller's messages to its devices, and have an interlock stop a heater.	Supervision and commands keep their intervals. The heater's stop reaches its device as soon as it does without the apps. The controller sends no device a message to carry out <code>system.read</code> or a subscription.
Extension methods	7.10.2	Read the documentation of each extension method that a controller serves, and send each from users of each role.	Each names an operation of Security and Access Table 6, is refused to roles that do not permit it, and is recorded when its operation is audited.
The controller	8.1.1	Send <code>system.describe</code> .	<code>specRevisions</code> lists each specification that the controller claims, with its revision, and <code>controller-api</code> with 0.1.
Devices	8.2.1, 8.2.3, 8.3.5	With a module, a network device, a bridged device, and built-in endpoints, subscribe to <code>system</code> . Disconnect the network device until the controller treats it as lost, and reconnect it; add a module, and remove one. Send <code>system.describeDevice</code> for each device and for <code>controller:self</code> .	<code>system.listDevices</code> lists every device, <code>controller:self</code> included, and the lost device with <code>lost true</code> until it returns. <code>system.devicesChanged</code> follows each change within 1 s. Each description is valid against the Device Classes schema, and is the one that the controller read.

Test	Requirements	Method	Pass criteria
Device state	8.3.3, 8.3.4	Subscribe to <code>system</code> for two devices. Switch an output, change a reading every 100 ms for 10 s, and raise a fault.	After the response, <code>system.state</code> arrives for each device and each of its endpoints. Each change then arrives within 1 s, no more than once a second for each endpoint, and each <code>system.state</code> carries every state attribute.
History	9.1.1, 9.2.2	Read a week of history of two systems, with each value of <code>include</code> , and with <code>parameters</code> and <code>eventTypes</code> , page by page, using each <code>next</code> up to 10 minutes after it was given. Validate each page.	Each page is a valid Data Model Document of revision 0.4, with every object that its records refer to. The records come in order of time, none twice, and each <code>next</code> is accepted.
Reading the configuration	10.1.1	Read the configuration document with and without <code>state</code> , and compare it with the controller's export.	It is complete and valid, the same as the export's, and has <code>state</code> only when asked.
Replacing the configuration	10.1.3, 10.1.4	Replace the configuration with a document that is not valid. While a feed mode, an override, and a program are active, replace it with a document without <code>state</code> that removes the feed mode and keeps the rest. Then replace it with a document whose <code>state</code> holds another override.	The first is refused with <code>invalid-document</code> and its problems, and the running configuration is unchanged. After the second, the feed mode has ended, and the override and the program continue. After the third, the saved state is the document's.
Status	10.2.1 to 10.2.3	Arrange outputs decided by each layer of Table 5, a virtual output, and a latched stop. Read the state, subscribe to <code>automation</code> , and then start a mode, end an override, and replace the configuration.	Every output, the virtual one included, has a status with the <code>layer</code> , <code>element</code> , <code>left</code> , <code>until</code> , <code>latched</code> , and <code>controlSource</code> that its situation gives, and the saved state is as Automation Section 17.8 defines it. <code>automation.changed</code> follows each change within 1 s, not more than once a second, and not for a change of <code>left</code> alone.
Modes	10.3.2, 10.3.3, 10.9.4	Start a feed mode, start it again, and start it again with <code>restart false</code> . Start and end the power-loss mode. Start a mode that the controller does not support. End an emergency mode as a user whose role permits <code>start-mode</code> but not <code>end-emergency-mode</code> , and then as one whose role permits it.	The second start moves the mode's end, and the third does not. The saved state names the session's user as the one who started it. The power-loss mode is refused with <code>not-by-person</code> , and the unsupported mode with <code>unsupported</code> . The emergency mode is refused with <code>not-permitted</code> , and then ended.
Scenes	10.4.1, 10.4.3	Start a photo scene with a <code>duration</code> shorter than its own, and then with one longer. End it. Let a scene trigger start a scene, and send <code>automation.endScene</code> .	The first starts, with <code>left</code> the <code>duration</code> given; the second is refused with <code>duration-too-long</code> . The person's scene fades out. The trigger's scene goes on.
Overrides	10.5.1, 10.5.2	Set an override with a level on an output whose class has no level, and then one with stop for 900 s. Read the state, and record the commands.	The first is refused with <code>invalid-state</code> . The second's <code>by</code> is the session's user, it expires after 900 s, and its command carries <code>revertAfter</code> .
Programs	10.6.2	Start a program with a <code>start</code> 2 minutes past; start a program twice; start a program that sets a function that a running program sets; restart the controller with its clock not valid, and start a program.	The requests are refused with <code>start-passed</code> , <code>program-conflict</code> , <code>program-conflict</code> , and <code>clock-invalid</code> .
Latches	10.7.2 to 10.7.4	Clear a latch whose <code>set</code> condition is true, and one whose <code>ownerReset</code> is <code>false</code> . Let a leak abort a water change, and clear it without <code>action</code> , then with <code>resume</code> while the leak lasts, and then after it ends. Clear a top-off unit's latched fault, then the same with the unit lost, and then with a unit that refuses <code>resetFault</code> .	<code>latch-set</code> ; <code>owner-reset-off</code> ; <code>action-needed</code> ; <code>cannot-resume</code> ; the water change resumes. The controller sends <code>resetFault</code> ; then <code>device-lost</code> ; then <code>device-refused</code> , with the unit's error.
Stopping everything	10.8.2 to 10.8.4	Start <code>stop-all</code> from a user of the <code>override</code> role, recording the sessions with devices; then in an	Each time, the controller begins sending the stops before it responds,

Test	Requirements	Method	Pass criteria
		emergency session; then while the clock is not valid, from a user whose grant has an expiry. Read the saved state each time.	the saved state names the session's user, and the request is carried out.
Matter bridges	10.9.1 to 10.9.3	From a Matter bridge's user, with <code>fabric</code> and <code>onlyOwn</code> , set an override of an output that has a person's override, and of one that has none; end a mode that a person started. Send <code>fabric</code> from another service's user. Read the audit log.	The first and the third are refused with <code>other-user</code> ; the second is carried out, and recorded with <code>via</code> <code>matter</code> and the fabric's <code>fabricIndex</code> , <code>label</code> , and <code>vendorId</code> . The other service is refused with <code>not-bridge</code> .
Following alarms	11.1.1, 11.1.2	Subscribe to <code>alarms</code> . Raise an alarm and a notice, raise a warning during a maintenance mode, and another in a recipient's quiet hours; then acknowledge, silence, clear, and close them. List the alarms.	The list holds every alarm and notice that is not closed, with the members of Alarms and Notifications Table 25. <code>alarms.changed</code> reports every change, the held warnings included, each as one message with one <code>id</code> and one <code>seq</code> .
Notifications through apps	11.2.1, 11.2.3	A user has two app installations: A, with a session subscribed with <code>notify true</code> and a push subscription, and B, with only a push subscription. Raise a critical alarm. Then let A's app stop responding, and raise another.	A receives <code>alarms.notify</code> , and B a push message. For the second alarm, A receives a push message after 10 s without a response.
Responding to notifications	11.2.2	Send an app <code>alarms.notify</code> while it runs in the background, and while it shows another screen.	The app responds within 5 s, before it shows the message.
Acknowledging	11.3.1, 11.3.2	Acknowledge two alarms, one of them acknowledged already; then send a list with an unknown <code>id</code> ; then acknowledge <code>all</code> .	<code>acknowledged</code> holds the first alarm only, which names the session's user. The request with the unknown <code>id</code> is refused with <code>unknown-alarm</code> , and acknowledges nothing. <code>all</code> acknowledges every alarm and notice that is not closed.
Tests	11.4.2	As a user of the <code>view</code> role, test the recipient that is the user, and then another recipient. As an owner, test a feed.	The first is carried out, and the second refused with <code>not-permitted</code> . The owner's test is carried out.
Push	11.5.2, 11.5.3	Register a push subscription from an installation twice, with different endpoints; register one in a session signed in with a password; then revoke the installation's credential, and raise an alarm.	The second subscription replaces the first. The password session is refused with <code>no-credential</code> . After the revocation, no push message goes to the installation's subscription.
Alarm settings	11.6.1, 11.6.3 to 11.6.5	Read the alarm settings, which have an email channel with a password and a webhook with a secret. As a user of the <code>view</code> role, change the quiet hours of the user's own recipient, and then a range. As a user of the <code>full-control</code> role, add a recipient. Replace them with a document that is not valid. Replace them with a document whose email channel has no password, and that adds a webhook channel without a secret. Read them again.	The document holds no password or secret, and <code>secrets</code> names both channels. The quiet hours are changed; the range and the recipient are refused with <code>not-permitted</code> . The invalid document is refused with <code>invalid-document</code> , and the settings kept. The email channel keeps its password; the new webhook's secret is in that result's <code>secrets</code> , and in no later result.
Users	12.1.1, 12.1.2	Send <code>users.list</code> as an owner, and as a user of the <code>view</code> role.	The owner receives every user, and the other user only itself. Neither result holds a password hash, a verifier, or a code.
Changing users	12.3.1	Give the last owner the <code>view</code> role; give a service the <code>owner</code> role; give an owner an expiry.	The requests are refused with <code>last-owner</code> , <code>service-owner</code> , and <code>owner-expiry</code> .
Removing users and credentials	12.3.2, 12.3.3	An owner has two credentials, each with an open session, a push subscription, and an entry in a device's pairing list; another user has signed in with a password. Revoke one of the owner's credentials. Remove the other user. Try to remove the last owner.	The revoked credential's session closes within 5 s, with code 4001, its push subscription is gone, and its key leaves the device's pairing list. The other user's session closes within 5 s. Removing the last owner is refused with <code>last-owner</code> .

Test	Requirements	Method	Pass criteria
Reading the audit log	12.4.1	Read the audit log with <code>after</code> and <code>limit</code> , and again until <code>more</code> is <code>false</code> .	Each response holds the records after <code>after</code> , in order, no more than <code>limit</code> , and <code>more</code> is <code>true</code> until the last.
Maintenance	13.1.1, 13.1.2, 13.1.4	Record a refill of a dosing head's container, with <code>left</code> ; a refill of a reservoir; a replaced reagent of an analyzer; an emptied waste container; maintenance of a skimmer with <code>task</code> ; a refill with <code>reagent</code> ; a replaced reagent of an analyzer that the controller has lost; and one of an analyzer that refuses <code>replaceReagent</code> .	The estimates are reset and the faults cleared as Table 8 says, and the events of Table 8 are recorded, the skimmer's with the task's <code>eventType</code> . The refill with <code>reagent</code> is refused with <code>invalid-record</code> ; the lost analyzer with <code>device-lost</code> ; and the other with <code>device-refused</code> .
Manual results	13.2.2 to 13.2.4	Enter a nitrate result of 10 mg/L-N, a calcium result of 4300 mg/L where 430 was meant, and a result in a unit that its parameter does not accept. Then enter the first two, with the calcium result <code>confirmed</code> .	The request with the wrong unit is refused with <code>invalid-result</code> , and nothing is recorded. Without the wrong unit, the calcium result is <code>needs-confirmation</code> , with the reason <code>range</code> , the nitrate result <code>checked</code> , and nothing is recorded. With the confirmation, both are recorded.
Laboratory results	13.3.2, 13.3.4 to 13.3.6	Import a laboratory document with a magnesium result outside its plausible range; then the same with its index in <code>confirmed</code> ; then the same document again. Import a document with an observation that belongs to no laboratory sample, and one that is not valid.	The first records nothing, and gives the magnesium result <code>needs-confirmation</code> . The second records every result. The third gives each <code>duplicate</code> , and records nothing. The others are refused with <code>not-laboratory</code> and <code>invalid-document</code> .
Export	14.1.1, 14.1.2	Export as an owner with <code>auditLog true</code> , and with <code>auditLog false</code> ; validate each file. Send <code>export.read</code> for the first file from another session.	Each file is a valid export file with a Document of the range, the configuration document with its saved state, and the alarm settings, and the audit log only in the first. The other session is refused with <code>unknown-file</code> .
Import	14.2.1, 14.2.2, 14.2.4 to 14.2.8	Upload an export file of 4 194 304 bytes to a second controller, with one part out of order; upload a file larger than the controller can hold; and upload a file with a member that would decompress to more. Check the first file. Import it as a user of the <code>full-control</code> role, when its alarm settings have recipients, and then as an owner.	The part out of order is refused with <code>out-of-order</code> , and the whole file is accepted. The larger file is refused in its first part with <code>file-too-large</code> and <code>limit</code> , and the other with <code>file-too-large</code> , before the controller has decompressed it. The check gives each member's operations. The import as <code>full-control</code> is refused with <code>not-permitted</code> , and as an owner imports everything: the configuration with its saved state and a new version, the alarm settings with a new version, and the history, with nothing twice. The audit log is <code>ignored</code> , and none of its records is in the second controller's audit log.
Requests that reach devices	7.11.1 to 7.11.4	Send <code>actions.feed</code> to a feeder that refuses it, to one that the controller has lost, to one that answers after 6 s, and to one under local control, recording the sessions with the devices; reconnect the lost feeder, and let the slow one report its state.	The first is refused with <code>device-refused</code> and the feeder's error; the second with <code>device-lost</code> , and no <code>feed</code> reaches the feeder when it returns; the third with <code>device-unanswered</code> within 5 s, and no second <code>feed</code> is sent before the controller has read the feeder's state and events; the fourth with <code>under-local-control</code> , and no command reaches the feeder.
Long operations	7.12.2	On one of two sessions, add a device, and install an update of another; close that session during a second <code>system.addDevice</code> .	<code>system.deviceAdded</code> and the last <code>updates.progress</code> , with the outcome, arrive on the session that started each, and on no other. The device of

Test	Requirements	Method	Pass criteria
			the second operation appears in <code>system.listDevices</code> once it is added.
Identifying	8.4.2	Identify a device, and one of its endpoints, while its outputs run.	The device and the endpoint show where they are, and no output changes.
Adding a device	8.5.2, 8.5.3, 8.5.5	With two units in pairing windows, with different discriminators, add one with its setup code. Add a device with an invitation that an owner's app made on it; take on a device whose owner's app added the controller to its pairing list; add one by address and port on another network segment; add one with <code>deviceNetwork true</code> ; and try to add one as a user of the <code>override</code> role. Record the sessions with the units, and read every result, notification, and audit record.	The controller tries the code only on the unit whose discriminator matches. Each device ends with every owner's key in its pairing list, under the controller's control, subscribed, and with each endpoint commanded; the last is on the controller's own network. The <code>override</code> user is refused with <code>not-permitted</code> . No code is in any result, notification, or audit record.
Removing a device	8.5.6, 8.5.7	With an app, remove a network device with <code>factoryReset true</code> , and a module, recording the sessions; then remove a device that the controller has lost.	The app offers to factory reset each device before it removes it. The controller releases each device, then resets it, then leaves its pairing list, in that order, and each endpoint takes its fail-safe behavior at once. Each device is forgotten, and its outputs raise <code>binding-missing</code> . Nothing is sent to the lost device.
Device settings	8.6.2 to 8.6.4	Write two config attributes of a module's endpoint, one of them out of range, and read its registers. Write a <code>failSafeTimeout</code> without <code>confirmed</code> , and then with it. Write <code>failSafeAction continue</code> to a return pump, and a program to a light.	The first write is refused, and neither register changes. The second is refused with <code>confirmation-needed</code> , and the third carried out, and recorded with the old and new values. The last two are refused with <code>protected-setting</code> .
Warnings of settings	8.6.5, 8.6.6	Check a change of an outlet's <code>loadCategory</code> to <code>heater</code> ; a <code>failSafeTimeout</code> of 600 s on an outlet on Wi-Fi that a heater's cutoff stops; and <code>failSafeAction continue</code> for a top-off unit whose water carries an additive. Then disable the backup thermostat of an outlet whose <code>failSafeAction</code> is <code>continue</code> , recording the writes.	The checks return <code>recommended</code> , the recommendation for a heater, and warnings of the safety function without a path, and of the top-off unit. The controller writes <code>failSafeAction state</code> and <code>failSafeState {"on": false}</code> before it disables the backup thermostat, as one confirmed change.
Restarting	8.7.1	Restart a doser while it doses, and again while it is idle.	The first is refused with <code>operation-running</code> ; the doser restarts after the second.
Passwords	12.5.1 to 12.5.3	Set passwords of 14 characters, of a common password of 16 characters, and of 64 characters with spaces and symbols, and read the controller's storage and every result. As a user of the <code>view</code> role, remove the user's own password while a session signed in with it is open; as an owner, remove another user's.	The first two are refused with <code>weak-password</code> , and the third accepted. Only a salted hash is stored, and no result holds the password. The session signed in with the password closes within 5 s.
Actions while held	15.1.2	Feed while a maintenance mode stops the feeder, and while <code>stop-all</code> lasts; dose while an interlock holds a dosing head with a latched stop, and while the head has <code>stuck-on</code> latched; recording the sessions.	Each is refused with <code>held</code> , nothing is sent, and the latched stop stays.
Tests on demand	15.1.4	Start a test on demand while the analyzer's waste is full, while the clock is not valid, and while its testing is suspended, and let the last give an accepted result.	The first is refused with <code>held</code> , and the second with <code>clock-invalid</code> . The third runs: the controller enables the analyzer for the test, and the accepted result ends the suspension.
Manual fills	15.1.5	Fill during a water change, and after the day's fill time is used up.	The first is refused with <code>held</code> , and the second with <code>limit-reached</code> .

Test	Requirements	Method	Pass criteria
Doses and primes	15.1.6	Give a dose that takes twice the head's <code>maxContinuousRun</code> ; prime for longer than <code>maxContinuousRun</code> ; and prime for longer than the priming time left in the day.	The dose is given in two parts, with a rest between. The primes are refused with <code>too-long</code> , with the longest as <code>limit</code> , and with <code>limit-reached</code> .
Following calibrations	16.2.1, 16.2.2	Start a probe's calibration at its port's own controls; restart the controller during a calibration hold; subscribe to <code>calibration</code> , and calibrate a probe.	Both calibrations are listed. <code>calibration.changed</code> follows each change of phase within 1 s, and not more than once a second.
Calibrating a probe	16.3.1 to 16.3.3, 16.3.6	Calibrate a pH probe with a temperature probe in the references, recording the functions' inputs and the compensation temperature written to the port. Try to start one while the clock is not valid. Calibrate an ORP probe with a reference of 231 mV against saturated silver chloride. Confirm that a probe is back before its calibration ends, and calibrate one whose reading shifts by more than its probe type's shift.	Both probes' inputs are unavailable before <code>beginCalibration</code> is sent; the reference's temperature is written to the port, and the system's again at the end. The start without a valid clock is refused with <code>clock-invalid</code> . The ORP point is recorded as 225 mV for a probe with a 3.5 mol/L silver chloride reference. The hold ends only after the calibration, the confirmation, and the settle time, and, after the shift, only once a person accepts the new reading.
Calibrating a dosing head	16.4.1, 16.4.3	Start a head's calibration while an interlock holds it, and with a <code>duration</code> longer than its <code>maxContinuousRun</code> . Run one, and complete it with a volume that changes its flow rate by 12 %, and then with <code>confirmed true</code> .	The first two are refused with <code>held</code> and <code>too-long</code> . The first completion returns <code>needs-confirmation</code> , with both rates, and sends nothing; the second sends <code>completeCalibration</code> .
Calibrating an analyzer	16.5.1	Start an analyzer's calibration while a maintenance mode holds it.	It is refused with <code>held</code> .
Corrections	16.6.1, 16.6.3, 16.6.4	With three comparisons of an analyzer's nitrate results with laboratory results, list the corrections, and apply the proposal from an app. Apply a correction that changes the latest result by 15 %, without <code>maxCorrection</code> , and then with <code>confirmed true</code> .	The list gives the comparisons and the proposal. The app shows the comparisons, the previous correction, and the change of the latest result before it applies one. The first correction is not applied, and returns <code>needs-confirmation</code> ; the second is applied, and the configuration document has a new version.
Firmware of every unit	17.1.1	With a module and a network device, consent to update checks, publish a release in a device's feed, and give the controller a manifest of another release. List the updates.	The controller, the module, and the device are each listed with what Security and Access Section 15.9.2 requires, and both releases are among those available.
Installing an update	17.2.1 to 17.2.3, 17.2.5	With an app, update a doser with a manifest signed by a key it does not have, one of a lower security version, and one of another model; then while it doses; then with a valid release, recording <code>updates.progress</code> .	The app shows the version, the channel, the notes, the security fixes, the interruption, and what the outputs do before it approves. The first three are refused with <code>manifest-refused</code> , and <code>firmware-refused</code> is recorded; the fourth with <code>operation-running</code> . <code>updates.progress</code> follows each change of state, and comes at least every 5 s while the image is sent.
Updating the controller	17.2.6	Check an update of the controller whose <code>interruption</code> is 120 s, while outputs have a <code>failSafeTimeout</code> of 60 s; install it, recording its supervision of its devices.	The check warns of the interruption. The controller supervises its devices while it writes the image, restarts only once the image is written and checked, and confirms the new firmware as Security and Access Section 15.4.2 says.
Update settings	17.3.2, 17.3.3	Turn automatic installation on before an owner has consented to update checks. Revert a device with no previous firmware, and another while it doses.	The requests are refused with <code>no-consent</code> , <code>no-previous</code> , and <code>operation-running</code> .

Test	Requirements	Method	Pass criteria
Remote access	18.1.1	Turn remote access off while an app is connected through a relay.	The controller's relay sessions, and the app's remote session, close within 5 s.
Emergency access	18.2.2, 18.2.3, 18.2.5	Read the emergency access as users of the <code>view</code> and <code>override</code> roles, and in an emergency session. Create two emergency codes, and read the controller's storage. Remove a code while an emergency session that it opened is open.	Only the <code>override</code> user outside the emergency session receives the passphrase. The codes have at least eight digits and different slots, appear only in their results, and only their verifiers are stored. The emergency session closes within 5 s.
Consent	19.1.1	List the consent of a new controller, and again after an owner has given one consent.	Each purpose has its description, destination, data, frequency, and retention, and <code>given</code> is <code>null</code> until an owner answers.
Erasing data	19.2.1	Erase data, and read the history, the alarm records, and the audit log.	The history and the alarm records are gone. The audit log is kept, with a record of the erasure.

21 Security and Privacy Considerations

1. This Specification carries the Device Protocol and the Security and Access specification, and their threats, protections, and remaining risks apply [OAP-DP Section 18; OAP-SA Sections 7 and 20].
2. **Least privilege.** Every method names the operation it needs, and reading is kept apart from changing: a house-sitter with the `view` role sees the system and sets their own quiet hours, and nothing more. Secrets never leave the controller: no result, audit record, or export file holds a password, a webhook secret after the result that made it, or a push subscription's keys.
3. **Apps show what people confirm.** The controller relies on the app to show the person what an imported document will do before it is used (Section 7.6.6). A dishonest app of a user with the `full-control` role could change the configuration anyway; what a configuration cannot change, fail-safe settings and the devices' own limits, stays the protection that does not depend on automation [OAP-AU Section 21.4], and every change is in the audit log.
4. **Races.** Versions keep two apps from undoing each other's changes unknowingly, and `onlyOwn` keeps a bridge from replacing a person's override.
5. **Matter bridges.** Only a Matter bridge's user may give a fabric, so that no other party can make its actions look like those of a home ecosystem in the audit log.
6. **Denial of service.** A controller limits what each session may hold, such as subscriptions and files, and never lets apps delay its control (Section 7.7.3). Reading the state of devices sends nothing to them, so that apps cannot slow a stop (Section 8.3.2).
7. **Untrusted files.** Export files and laboratory documents come from outside the controller: it validates each fully before it uses any part of it, checks sizes before it decompresses, and imports all or nothing (Section 14.2).
8. **Privacy.** History, configurations, alarm settings, and the audit log hold the owner's data, such as names, schedules, and where the system is [OAP-AU Section 21.6]. The methods reveal them only to users whose roles permit it, and the export file is the owner's own copy, which leaves the controller only for the session that asked for it. A push subscription's endpoint names the push service that the user chose; push messages are encrypted from end to end [OAP-AN Section 17.2.3].
9. **Emergency sessions and the clock.** An emergency session can only operate the system for a while, and a user whose grant has an expiry can only stop equipment while the controller's clock is not valid (Sections 7.2.3 and 7.2.4).

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	
A controller and an app from different implementations have worked together through every namespace.	Not started	
The implementations have passed the test procedures in Section 20.4, and those of the Device Protocol and Security and Access specifications.	Not started	
Every normative reference is publicly available.	Done	All are free to read.
The method definitions, schema, and examples are published, and the examples pass the validator.	Done	
The Open Issues appendix is empty, or each open issue is explicitly deferred.	In progress	Issue 1 waits on the bench tests of the breadboard controller.
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 — Message Exchanges (Informative)

The `examples` directory holds complete exchanges, which the validator checks:

- `alarms.json`: an owner's app follows the alarms, receives a critical alarm with `alarms.notify`, acknowledges and silences it, registers for push, and sends a test; a house-sitter with the `view` role cannot acknowledge.
- `modes-and-overrides.json`: an owner reads the state, starts a feed mode and a photo scene, overrides a pump, starts a treatment program, clears a latch, and stops everything, and a user with the `view` role is refused.
- `configuration.json`: an owner reads, checks, and replaces the configuration document and the alarm settings, and loses a race to another app, which `ifVersion` catches.
- `users.json`: an owner invites a house-sitter, changes a role, revokes a lost phone, removes a user, and reads the audit log, and the controller refuses to remove its last owner.
- `history-and-records.json`: an app reads a day of history in pages, records a refill, enters manual results, one of which needs confirmation, and imports a laboratory's results.
- `export.json`: an owner exports a controller's data, and imports it into a new controller.
- `matter-bridge.json`: a Matter bridge on another host follows the state, stops a heater for a home app, is refused where a person's override is in effect, and starts a feed mode without moving its end.
- `devices.json`: an owner adds a doser with its setup code, which the controller reports when it is added, checks and confirms a change of its fail-safe settings, identifies it, restarts it, and removes another device, with a factory reset.
- `actions-and-calibrations.json`: an owner feeds the fish, starts a test on demand, checks and gives a manual dose, is refused a prime that is too long, calibrates a pH probe, with the probe's hold ending once the owner confirms it is back, calibrates a dosing head, and applies a correction to an analyzer's results.
- `owner-settings.json`: an owner lists the firmware of the controller and its devices, checks and installs an update of a doser, following its progress, sets the update channel, turns remote access on, creates an emergency code for a neighbor, answers a request for consent, sets a password, labels an app, and erases the history.

Cryptographic values in the examples are random placeholders of the right length.

Appendix C — Open Issues (Informative)

Issue 1 waits on the bench tests. Issue 2 records the choices that revision 0.1 made, and the Project Lead's decisions on its scope, which the Project Lead accepted and made on 18 October 2026. Each issue says how a revision resolves it, or why it stays open. Issue numbers do not change, because other specifications may refer to them.

1. **Numbers.** *Open: waits on the bench tests of the breadboard controller.* The 1 048 576-byte messages and the 1 000 000-byte documents that controllers accept (Section 7.5), the 4 194 304-byte uploads (Section 14.2.2), the 10 minutes for which cursors and files stay usable (Sections 9.2.2 and 14.1.2), the 1 s within which notifications follow a change (Sections 8.3.3, 8.3.5, 10.2.3, and 16.2.2), and the 4 s that a controller waits for a device's answer (Section 7.11.3) need confirming on the reference controller.
2. **Choices of revision 0.1.** *Resolved in revision 0.1: the Project Lead accepted each choice, and decided the scope, on 18 October 2026.* The Project Lead decided that revision 0.1 covers, besides the seven parts that the scope of revision 1.0 of the specifications gave it, `system.identify`, which a Matter bridge on another host needs for a home app's Identify [OAP-MM Table 8], and every other operation of Security and Access Table 6 that apps need: actions on demand, calibrations, device and fail-safe settings, adding and removing devices, firmware updates, restarts, remote and emergency access, consent, erasing data, factory reset, passwords, and the labels of credentials (Sections 8, 12.5, and 15 to 19). Managing a Matter bridge, allowing firmware that a manufacturer did not sign, turning owner recovery off, and signing out other sessions stay outside it (Section 3). Revision 0.1 made these choices where its scope left them open:
 - controllers accept messages of 1 048 576 bytes, and documents of up to 1 000 000 bytes, so that every document travels whole in one message, and only export files and firmware images go in parts (Sections 7.5, 14.2, and 17.2);
 - errors use only the codes of the Device Protocol, with a reason of Table 2 in their `data`, so that the Device Protocol needs no change (Section 7.4);
 - the controller delivers a message of the `app` channel with `alarms.notify`, a request that the app answers, so that the answer is delivery, and uses push after 10 s without one (Section 11.2);
 - a Matter bridge on another host gives its fabric with `fabric`, accepted only from a bridge's user, and uses `onlyOwn` and `restart false`, so that it never replaces a person's override or moves a mode's end, even in a race (Section 10.9);
 - documents have versions, and are replaced with `ifVersion`; checks change nothing; and an app shows a document from elsewhere, and has the person confirm it, before it replaces one (Section 7.6);
 - a configuration document without `state` keeps the saved state of the elements that remain, and ends the rest (Section 10.1.4);
 - an export file is imported all or none, its audit log never, and its history needs `enter-results` (Section 14.2);
 - results and laboratory documents are recorded all or none for each request, with a check first, and laboratory vessels are mapped by `vessels` (Sections 13.2 and 13.3);
 - invitations stay the Device Protocol's `pairing.invite` (Section 12.2);
 - `system.addDevice` and `updates.install` answer at once, and report the outcome to the session that started them, and other requests that reach a device wait at most 4 s for its answer, and then answer `device-unanswered`, so that no response waits longer than the Device Protocol allows (Sections 7.11 and 7.12);
 - a change of fail-safe settings carries `confirmed true`, which the app sends only once the person has confirmed it (Section 8.6.3);
 - each emergency code is a user of its own, the emergency link's passphrase goes only to users of the `override` role or above outside emergency sessions, and erasing data keeps the configuration, the alarm settings, the users, and the audit log (Sections 18.2 and 19.2); and
 - actions and calibrations name a device and an endpoint, so that a head can be primed or calibrated before the configuration binds it; modules join by themselves, with no step to accept them; a device is factory reset as it is removed; and one set of calibration methods serves probes, dosing heads, and analyzers (Sections 8.5, 15, and 16).