



OPEN AQUARIUM PROJECT SPECIFICATION

Security and Access

PRE-DRAFT

Revision	0.3
Status	Pre-Draft. A candidate document submitted for consideration.
Date	2026-10-18
Identifier	security-and-access
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 `security-and-access` 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 permissions file, JSON Schemas, and examples 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 permissions file, schemas, and examples under the Apache License 2.0. Every mechanism it requires is defined by a free, public standard, mostly IETF RFCs, and can be built with open-source libraries. No certificate authority, manufacturer attestation, membership, or fee stands between a maker and a working product: trust is set up by the owner, not issued by anyone. Equipment that an owner builds is a first-class participant. Devices are expected to let their owners run firmware of their own, and the reference designs must (Section 15.8).

2.2 Interoperability

An app from one maker can pair with a controller from another, and a controller can pair with, update, and command devices from any maker, because pairing, secure channels, roles, update manifests, and audit records are defined once for everyone. A controller verifies any maker’s update manifests the same way, so it can keep a mixed system up to date. The Specification reuses TLS 1.3, SPAKE2+, JSON Web Signature, and the identifiers and terms of the other Open Aquarium specifications.

2.3 Local Control

Pairing, authentication, roles, updates, and emergency access all work on the local network with no internet connection, cloud account, or vendor. Remote access is optional, off by default, and possible without any maker’s service (Section 13). Nothing leaves the owner’s network without the owner’s consent, except what the system needs to work locally (Section 16). Owners can export their history, configuration, and audit log at any time.

2.4 Safety

Security never stands between the equipment and its fail-safe behavior. Stopping equipment needs the smallest role that can change anything, and emergency access lets a person stop or override equipment when the network is down (Section 14). Losing a secure channel is losing contact, which the Fail-Safe Behavior specification already makes safe. Authentication does not depend on the clock, so an invalid clock or an expired certificate can never stop control (Section 10.7). `stop-all` stops every output at once, except those that signal alarms, which it leaves to their alarm function, with stops that end by themselves even if the controller fails, within a day at most, and nothing delays or refuses them (Section 11.8). Every safety function keeps a path that uses no radio link, and a controller warns its owners while one has none (Section 10.9). Firmware updates never start while equipment is dosing or filling, and a failed update returns to working firmware by itself (Section 15). Who may change fail-safe settings is controlled, and every change is recorded (Sections 11 and 12).

2.5 Sustainability

Because nothing depends on a maker's cloud, equipment keeps working, and stays manageable, after its maker stops supporting it or goes out of business. Manufacturers publish how long they will provide security updates (Section 18.2), and owners can keep updating offline from files they have saved. Factory reset and ownership transfer let equipment be sold and reused safely (Section 9.9).

Revision History

Revision	Date	Description	Authors
0.1	2026-10-14	Initial Pre-Draft.	Trent Scholl
0.2	2026-10-16	Safety functions as Fail-Safe Behavior Section 13.4 defines them, each with a path that uses no radio link, and a standing warning, not a refusal, while one has none; Bluetooth for setup only, with emergency access over Bluetooth removed; the audit events <code>fabric-added</code> and <code>fabric-removed</code> for each Matter fabric added to or removed from a controller's bridge, however it happened, with every owner notified; <code>start-mode</code> widened, as the Project Lead decided, to start a mode of any kind, end one other than an emergency mode, and start or end a lighting scene; <code>stop-all</code> sent with <code>revertAfter</code> except for the stops that end an operation, which never resumes, with dosing heads that run their own program held, and with <code>COMMAND_EMERGENCY_STOP</code> only for a module that has not accepted its stops, and never delayed; the 15-minute limit only for windows of a setup code, and the window for new network settings listed; emergency access on a controller's device network only with its clients isolated and only while emergency access is available (provisional); emergency access available while the home network, rather than the local network, is down; <code>open-pairing-window</code> only on devices, with invitations on a controller under <code>manage-users</code> and <code>manage-own-credentials</code> ; taking control as the Device Protocol defines it, with endpoints entering their fail-safe behavior at a new controller; the device attributes of Device Classes revision 0.2 for firmware information, with <code>org.openaquariumproject.firmware</code> read only from devices of revision 0.1, and changes of the update channel only with <code>install-firmware</code> ; the user member <code>grantExpires</code> , as the Device Protocol names it; Prover, Verifier, pairing verifier, unit, home network, and device network defined; claiming recorded on a device only if it keeps an audit log; replacing a controller whose devices were on its device network; the Device Protocol, Alarms and Notifications, and Matter Mapping specifications as references; test procedures for every requirement, each naming the requirements it checks; references cited by tag; one sentence per line; the decided issues resolved, and every other issue given a status; the Project Lead's decisions of 16 October 2026 applied: emergency access on a device network kept as restricted, and no longer provisional, and the module bus's emergency stop kept as a fallback, with no change to the Module Bus.	Trent Scholl
0.3	2026-10-18	The Project Lead's decisions of 17 October 2026 applied: the operations <code>record-maintenance</code> , for records that change no output, and <code>enter-results</code> , which only <code>full-control</code> permits, for entering test results and importing laboratory documents; <code>stop-all</code> leaves the outputs that an alarm function drives to that function, and no longer asks for an order of the stop and the hold of a dosing head; a browser reaches a unit's web interface through a paired app, which listens on the loopback address and relays its requests over a connection authenticated with its identity key; one service user for	Trent Scholl

Revision	Date	Description	Authors
		each Matter bridge, whose role caps every fabric, records of the bridge's actions with <code>via matter</code> and the fabric in <code>details</code>, and turning a bridge on, opening its commissioning window, and removing a fabric with <code>manage-users</code>; TLS 1.3 for every connection, battery devices on Thread included, with DTLS removed; SPAKE2+, the relay as the default for remote access, owner-installed firmware, roles, and notification operations no longer provisional. Also: a maximum expiry of 24 hours for <code>stop-all</code> (provisional), whose stops end no latched stop, and <code>device.stop</code> and <code>COMMAND_EMERGENCY_STOP</code> never sent to a device or module with an output that an alarm function drives; emergency codes accepted with no pairing window only over the emergency link while it is available, and every session over that link an emergency session; <code>backupThermostat</code> among the fail-safe settings; [OAP-LP] a normative reference; "A pass criterion that checks what a SHOULD asks applies to a product that does it"; a test for each new requirement; the schemas' <code>\$id</code> <a href="https://openaquariumproject.org/schemas/<name>/0.3">https://openaquariumproject.org/schemas/<name>/0.3; references to the final revisions of the other specifications; issues 10, 15 to 22, 26, and 27 deferred beyond revision 1.0.	

Contents

1	License	2
2	Compliance with Open Aquarium Tenets	2
2.1	Openness	2
2.2	Interoperability	2
2.3	Local Control	2
2.4	Safety	3
2.5	Sustainability	3
	Revision History	3
3	Scope	8
4	References	8
4.1	Normative References	8
4.2	Informative References	9
5	Terms, Definitions, and Conventions	10
5.1	Requirements Language	10
5.2	Terms and Definitions	10
5.3	Notation	11
6	Introduction	12
6.1	Purpose	12
6.2	Design Principles	12
6.3	Overview	13
7	Threat Model	14
7.1	Assets	14
7.2	Threats	14
7.3	Out of Scope	15
7.4	Trust Assumptions	15
7.5	Security Objectives	15
8	Identity and Keys	15
8.1	Identity Keys	15
8.2	Identity Certificates	16
8.3	Key IDs	16
8.4	Storing Secrets	16
8.5	Identity and Privacy	16
9	Setup and Pairing	16
9.1	No Default Credentials	16
9.2	Setup Codes	17
9.3	Pairing Windows	17
9.4	Pairing Protocol	17
9.5	Claiming	18
9.6	Bluetooth During Setup	18
9.7	Modules	18
9.8	Factory Reset	18
9.9	Transferring Ownership	19
9.10	Physical Presence	19
10	Secure Channels on the Local Network	19
10.1	TLS 1.3	19
10.2	TLS Profile	19
10.3	Authenticating the Other Party	20
10.4	Web Interfaces	20
10.5	Passwords	21
10.6	Revocation	21
10.7	Time	21
10.8	Network Exposure	21
10.9	Resisting Denial of Service	21
11	Users, Roles, and Sharing	22
11.1	Users	22
11.2	Roles	22
11.3	Operations	22

11.4	Grants and Expiry	25
11.5	Sharing	25
11.6	Device Pairing Lists	25
11.7	Revocation and Recovery	26
11.8	Stopping Everything	26
12	Audit Log	27
12.1	What Is Recorded	27
12.2	Records	27
12.3	Keeping and Exporting the Log	28
13	Remote Access	28
13.1	Requirements	28
13.2	Architectures	29
13.3	Relays	29
13.4	Cloud Services and Integrations	29
14	Emergency Local Access	30
14.1	Purpose	30
14.2	Paths	30
14.3	When It Is Available	30
14.4	Credentials	30
14.5	Limits	30
14.6	The Direct Link	31
15	Firmware Updates	31
15.1	Signed Images	31
15.2	Verification on the Device	31
15.3	Anti-Rollback	31
15.4	Keeping the Previous Firmware	32
15.5	Installing Updates	32
15.6	Update Manifests and Feeds	32
15.7	Channels	33
15.8	Owner-Installed Firmware	33
15.9	Firmware Information	34
15.10	Manufacturers' Keys	34
16	Privacy	34
16.1	Principle	34
16.2	What May Leave the Owner's Network	35
16.3	Consent	35
16.4	What Telemetry May Contain	35
16.5	Network Destinations	35
16.6	Export and Erasure	35
17	The Module Bus	36
17.1	Physical Access Model	36
17.2	What the Model Does Not Cover	36
17.3	Requirements in This Revision	36
17.4	Adding Message Authentication on CAN FD	36
18	Manufacturer Obligations	37
18.1	Vulnerability Handling	37
18.2	Support Period	37
18.3	Documentation	38
19	Conformance	38
19.1	Devices	38
19.2	Controllers	38
19.3	Apps	38
19.4	Manufacturers	38
19.5	Machine-Readable Definitions and Tools	38
19.6	Test Procedures	38
20	Security Considerations	44
Appendix A — Approval Checklist		44
Appendix B — Flows (Informative)		45
B.1	Claiming a Controller and Adding a Device	45

B.2	Sharing with a House-Sitter	45
B.3	A Lost Phone	45
B.4	Replacing a Failed Controller	46
Appendix C — Consumer IoT Security Provisions (Informative)		46
Appendix D — Open Issues (Informative)		46

3 Scope

This Specification defines:

- a threat model for home aquarium systems;
- the identity keys of devices, controllers, and apps;
- how an owner claims a new device or controller, without default passwords, and pairs apps and devices with it;
- authentication and encryption on the local network, without public certificate authorities;
- users, roles, and permissions, with sharing, expiry, and revocation;
- the audit log of changes;
- requirements and architectures for remote access without a maker's cloud;
- emergency local access when the network is down;
- firmware updates: signing, verification, rollback, channels, the owner's consent, and update manifests;
- privacy: what may leave the owner's network, consent, export, and erasure;
- the security model of the module bus, and how message authentication could be added to it; and
- what manufacturers publish about vulnerabilities and support periods.

This Specification does not define:

- message encodings, discovery, or how pairing messages are carried, which the Device Protocol [OAP-DP] and Module Bus [OAP-MB] specifications define;
- the relay protocol for remote access, which the Device Protocol specification defines [OAP-DP Section 15.4];
- how alarms are delivered, which the Alarms and Notifications specification defines [OAP-AN];
- the security of Matter, which a Matter bridge uses as Matter defines it (Section 13.4) [OAP-MM];
- the physical security of enclosures and cabinets; or
- legal obligations, such as those of the Cyber Resilience Act [CRA] and the United Kingdom's product security regulations [PSTI], although it is designed to help makers meet them.

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 9846]** Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 9846, July 2026.
- **[RFC 9266]** Whited, S., "Channel Bindings for TLS 1.3", RFC 9266, July 2022.
- **[RFC 5280]** Cooper, D., et al., "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, May 2008.
- **[RFC 9383]** Taubert, T. and C. Wood, "SPAKE2+, an Augmented Password-Authenticated Key Exchange (PAKE) Protocol", RFC 9383, September 2023.
- **[RFC 5869]** Krawczyk, H. and P. Eronen, "HMAC-based Extract-and-Expand Key Derivation Function (HKDF)", RFC 5869, May 2010.
- **[RFC 8018]** Moriarty, K., Ed., Kaliski, B., and A. Rusch, "PKCS #5: Password-Based Cryptography Specification Version 2.1", RFC 8018, January 2017.
- **[RFC 4648]** Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, October 2006.
- **[RFC 7515]** Jones, M., Bradley, J., and N. Sakimura, "JSON Web Signature (JWS)", RFC 7515, May 2015.
- **[RFC 7517]** Jones, M., "JSON Web Key (JWK)", RFC 7517, May 2015.
- **[RFC 7518]** Jones, M., "JSON Web Algorithms (JWA)", RFC 7518, May 2015.
- **[RFC 7638]** Jones, M. and N. Sakimura, "JSON Web Key (JWK) Thumbprint", RFC 7638, September 2015.
- **[RFC 8037]** Liusvaara, I., "CFRG Elliptic Curve Diffie-Hellman (ECDH) and Signatures in JSON Object Signing and Encryption (JOSE)", RFC 8037, January 2017.

- **[RFC 9864]** Jones, M.B. and O. Steele, "Fully-Specified Algorithms for JSON Object Signing and Encryption (JOSE) and CBOR Object Signing and Encryption (COSE)", RFC 9864, October 2025.
- **[RFC 8259]** Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, December 2017.
- **[RFC 3339]** Klyne, G. and C. Newman, "Date and Time on the Internet: Timestamps", RFC 3339, July 2002.
- **[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.
- **[FIPS 186-5]** National Institute of Standards and Technology, "Digital Signature Standard (DSS)", FIPS PUB 186-5, February 2023.
- **[SemVer]** Preston-Werner, T., "Semantic Versioning 2.0.0", <https://semver.org/spec/v2.0.0.html>.
- **[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-MB]** Open Aquarium Project, "Open Aquarium Module Bus", Revision 0.4, <specs/module-bus/spec.md>.
- **[OAP-AU]** Open Aquarium Project, "Open Aquarium Automation", Revision 0.3, <specs/automation/spec.md>.
- **[OAP-DP]** Open Aquarium Project, "Open Aquarium Device Protocol", Revision 0.3, <specs/device-protocol/spec.md>.
- **[OAP-AN]** Open Aquarium Project, "Open Aquarium Alarms and Notifications", Revision 0.3, <specs/alarms-and-notifications/spec.md>.
- **[OAP-MM]** Open Aquarium Project, "Open Aquarium Matter Mapping", Revision 0.3, <specs/matter-mapping/spec.md>.
- **[OAP-LP]** Open Aquarium Project, "Open Aquarium Lighting Programs", Revision 0.3, <specs/lighting-programs/spec.md>.

4.2 Informative References

- **[P0001]** Open Aquarium Project, "Proposal 0001: Reference Controller Architecture", <proposals/0001-reference-controller-architecture.md>.
- **[ETSI 303 645]** ETSI, "Cyber Security for Consumer Internet of Things: Baseline Requirements", ETSI EN 303 645 V3.1.3, September 2024.
- **[CRA]** Regulation (EU) 2024/2847 of the European Parliament and of the Council on horizontal cybersecurity requirements for products with digital elements (Cyber Resilience Act).
- **[PSTI]** The Product Security and Telecommunications Infrastructure (Security Requirements for Relevant Connectable Products) Regulations 2023, UK SI 2023/1007.
- **[NIST 800-63B]** National Institute of Standards and Technology, "Digital Identity Guidelines: Authentication and Authenticator Management", SP 800-63B-4, July 2025.
- **[RFC 8449]** Thomson, M., "Record Size Limit Extension for TLS", RFC 8449, August 2018.
- **[RFC 7250]** Wouters, P., Ed., et al., "Using Raw Public Keys in Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", RFC 7250, June 2014.
- **[RFC 9147]** Rescorla, E., Tschofenig, H., and N. Modadugu, "The Datagram Transport Layer Security (DTLS) Protocol Version 1.3", RFC 9147, April 2022.
- **[RFC 8613]** Selander, G., et al., "Object Security for Constrained RESTful Environments (OSCORE)", RFC 8613, July 2019.
- **[RFC 9528]** Selander, G., Preuß Mattsson, J., and F. Palombini, "Ephemeral Diffie-Hellman Over COSE (EDHOC)", RFC 9528, March 2024.
- **[RFC 7914]** Percival, C. and S. Josefsson, "The scrypt Password-Based Key Derivation Function", RFC 7914, August 2016.

- **[RFC 9106]** Biryukov, A., et al., “Argon2 Memory-Hard Function for Password Hashing and Proof-of-Work Applications”, RFC 9106, September 2021.
- **[CPace]** Abdalla, M., Haase, B., and J. Hesse, “CPace, a balanced composable PAKE”, Internet-Draft draft-irtf-cfrg-pace, work in progress.
- **[RFC 4493]** Song, JH., et al., “The AES-CMAC Algorithm”, RFC 4493, June 2006.
- **[RFC 8520]** Lear, E., Droms, R., and D. Romascanu, “Manufacturer Usage Description Specification”, RFC 8520, March 2019.
- **[RFC 6762]** Cheshire, S. and M. Krochmal, “Multicast DNS”, RFC 6762, February 2013.
- **[RFC 6763]** Cheshire, S. and M. Krochmal, “DNS-Based Service Discovery”, RFC 6763, February 2013.
- **[RFC 8915]** Franke, D., et al., “Network Time Security for the Network Time Protocol”, RFC 8915, October 2020.
- **[RFC 8291]** Thomson, M., “Message Encryption for Web Push”, RFC 8291, November 2017.
- **[RFC 8554]** McGrew, D., Curcio, M., and S. Fluhrer, “Leighton-Micali Hash-Based Signatures”, RFC 8554, April 2019.
- **[Cyphal]** OpenCyphal, “Cyphal Specification”, Version 1.0, revision 2025-05-16, https://specification.opencyphal.org/Cyphal_Specification.pdf.
- **[MCUboot]** MCUboot project, “MCUboot design”, <https://docs.mcuboot.com/design.html>.

5 Terms, Definitions, and Conventions

5.1 Requirements Language

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

5.2 Terms and Definitions

Terms defined in the Device Classes specification [OAP-DC], such as device, endpoint, controller, and binding, in the Fail-Safe Behavior specification [OAP-FS], such as contact, fail-safe behavior, and safety function, in the Automation specification [OAP-AU], such as override and mode, and in the Device Protocol specification [OAP-DP], such as session, have the same meaning here.

Owner: The person, or people, to whom a system belongs, and who hold the `owner` role on it (Section 11.2).

System: The equipment one owner manages together: usually one controller and its devices.

App: Software that people use to see and control a system, such as a phone app, a desktop program, or a web interface. Each installation of an app on one phone or computer is a separate party.

Party: Anything that authenticates itself: a device, a controller, an app installation, or a service.

Unit: A device or a controller: a party that accepts sessions and that an owner claims.

Service: A party that acts for the owner without a person, such as a home automation server or a cloud service.

Identity key: The key pair with which a party proves who it is (Section 8.1).

Key ID: The SHA-256 hash of a party’s public identity key (Section 8.3).

User: An entry in a controller’s user list: a person or a service, with a role and one or more credentials (Section 11.1).

Credential: What a user presents to prove who they are: the identity key of one of their app installations, a password, or an emergency code.

Role: A set of permissions: `view`, `override`, `full-control`, or `owner` (Section 11.2).

Operation: An action that a role permits, listed in Table 6.

Grant: A user’s role, together with its expiry and whether it may be used remotely.

Pairing: Establishing trust between two parties with a shared code, after which each knows the other's identity key (Section 9.4).

Claiming: The first pairing of a device or controller, which makes the party that pairs with it its owner (Section 9.5).

Setup code: A secret code that comes with a device or controller and is used to claim it (Section 9.2).

Prover: The party in a pairing exchange that knows the code, such as the owner's app claiming a device, or a controller adding one (Section 9.4.1).

Verifier: The party in a pairing exchange that holds a verifier for the code, and checks the code with it, such as a device being claimed (Section 9.4.1). Written with a capital, "the Verifier" means this party.

Pairing verifier: What a unit stores so that it can check a code during pairing without storing the code itself: the SPAKE2+ values `w0` and `L`, with their salt and iteration count (Section 9.4). Written in lowercase, "a verifier" means these values.

Invitation: A one-time code with which an owner shares access with a new user, or a user adds a new app installation (Section 11.5).

Pairing window: A period during which a device or controller accepts pairing (Section 9.3).

Pairing list: The identity keys that a device trusts, each with a role (Section 11.6).

Physical presence: Proof that a person is at a unit: pressing a button on it, or reading a code from its display (Section 9.10). Power-cycling a unit is not physical presence, because a smart plug can do it remotely.

Secure channel: A connection protected as Section 10 requires.

Relay: A server that forwards encrypted connections between apps and a controller that cannot reach each other directly, without being able to read them (Section 13).

Emergency access: Limited local access to a controller over a direct link while the home network is down (Section 14).

Emergency code: A code that lets a person without a paired app use emergency access (Section 14.4).

Local network: The IP networks in the owner's home that the owner's equipment uses, including a network the controller hosts for its devices, as opposed to the internet.

Home network: The local network other than a device network.

Device network: A Wi-Fi network that a controller hosts for its devices [OAP-DP Section 14.1].

Firmware image: The software a device runs, as its manufacturer builds and signs it.

Update manifest: A signed description of one firmware release (Section 15.6).

Update key: A key with which a manufacturer signs update manifests (Section 15.6).

Update feed: The list of a model's releases that its manufacturer publishes (Section 15.6).

Security version: A number in each firmware image that a device never goes below (Section 15.3).

Channel: Whether a release is for general use (`stable`) or for testing (`beta`), and which of them a device installs, its `updateChannel` (Section 15.7).

Secure boot: Verification of firmware at every start by code that cannot be changed, with keys fixed in hardware.

Telemetry: Any data about a system that a device, controller, or app sends to anyone other than the owner's own parties.

5.3 Notation

Role names, operation names, JSON members, and the names of attributes, commands, and faults appear in `monospace`. Durations are in seconds unless stated. References to other Open Aquarium specifications name them, as in "Fail-Safe Behavior Section 7.5", or use their tags from Section 4, as in "[OAP-FS Section 7.5]". A section number with a third part refers to a numbered item of that section: "Section 9.2.3" means item 3 of Section 9.2.

6 Introduction

6.1 Purpose

Aquarium controllers can boil, starve, or poison what lives in the tank, and flood the room they stand in. Yet commercial products protect them poorly. Table 1 lists weaknesses of four widely used commercial controller ecosystems and of other connected products, and where this Specification responds.

Table 1 — Weaknesses of commercial products

Finding	Response	Section
A local interface protected by a default password, over plain HTTP.	No default credentials; every connection encrypted and authenticated.	9.1, 10
Local interfaces with no authentication at all.	Every party authenticates on every connection.	10
Settings, alerts, or quick actions that work only through the maker's cloud.	Everything works locally; remote access without a maker's cloud.	13
No published account of how firmware is signed.	Signed images, verified on the device, with rollback.	15
Roles in only one ecosystem.	Four roles, sharing with expiry, and revocation.	11
No remote access without the maker's cloud, anywhere.	Owner-run relays, VPNs, and relay services with end-to-end encryption.	13
Data sent to the maker by default.	Opt-in consent for all telemetry.	16

6.2 Design Principles

1. **No shared secrets.** No device ships with a password, key, or code that another unit shares.
2. **The owner sets up trust.** Parties trust each other because the owner paired them, with a code, on the local network. No certificate authority or account is involved.
3. **One way to authenticate.** Local, remote, and emergency access use the same secure channels, credentials, and roles.
4. **Least privilege.** Each user, app, and service gets the smallest role that does its job, and each role is defined by the operations it permits.
5. **Security never makes equipment less safe.** When a security mechanism fails, equipment loses contact and takes its fail-safe behavior. Nothing in this Specification stops a controller from stopping equipment.
6. **The owner decides.** Who has access, when firmware is updated, whether remote access is on, and what leaves the network are the owner's choices.
7. **Fits a microcontroller.** Every requirement can be met by a controller built on a common single-chip Wi-Fi microcontroller, and the module requirements by a smaller microcontroller.
8. **The owner stays in control of the hardware.** Physical presence always lets the owner reset and reclaim equipment.

6.3 Overview

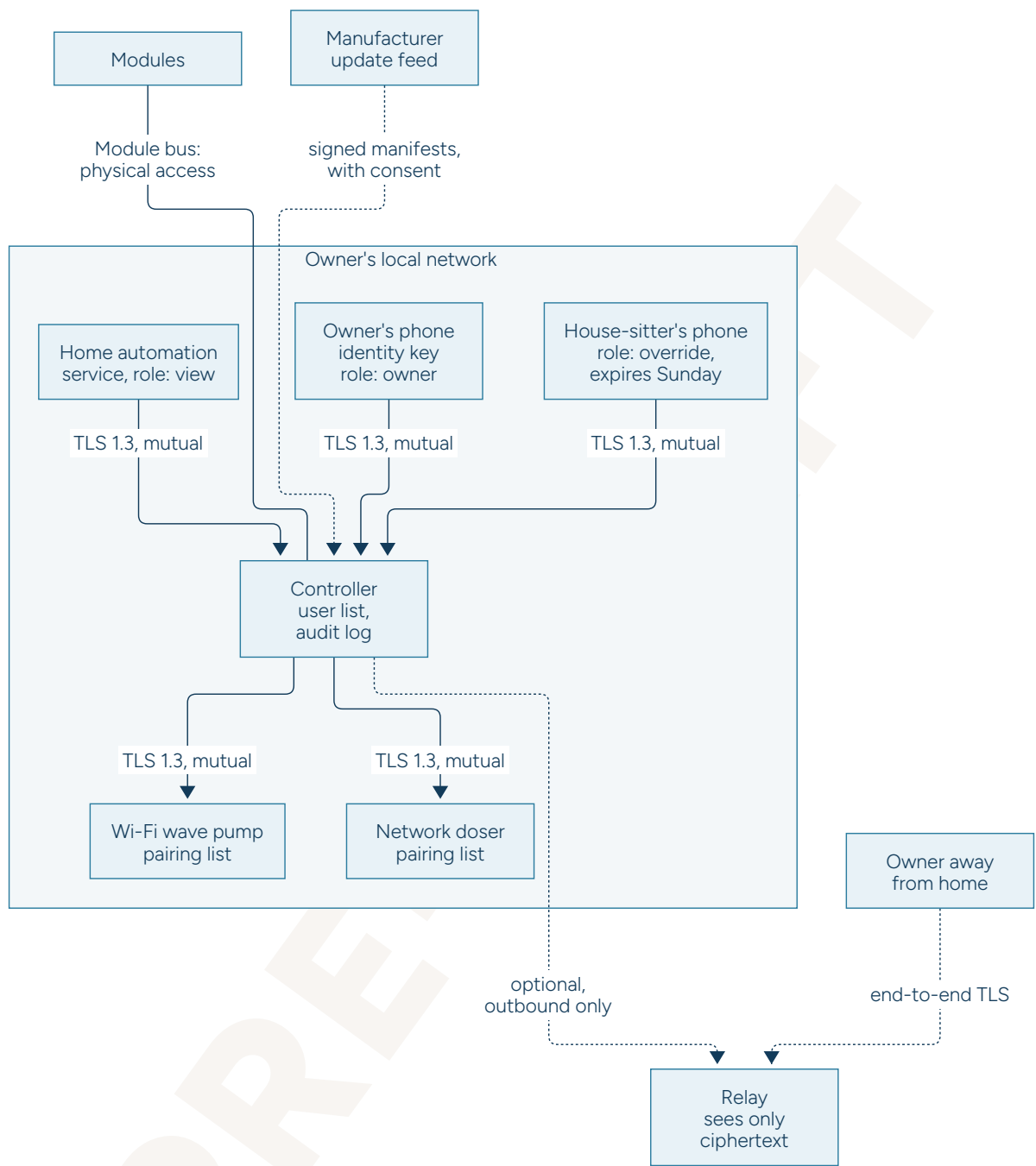


Figure 1 — Parties and secure channels in a system

Apps, services, and network devices talk to the controller over TLS 1.3, and each side checks the other's identity key against the keys the owner paired (Sections 8 to 10). The controller holds a list of users and their roles, and enforces the roles (Section 11). Each network device trusts only the keys in its pairing list, normally the controller's and the owners' (Section 11.6). Modules on the module bus rely on physical access control (Section 17).

Table 2 — Mechanisms of this Specification

Need	Mechanism	Section
Know who a party is	An identity key generated on each party; its key ID	8
Set up trust with no default password	A setup code with each unit, and pairing, with SPAKE2+	9

Need	Mechanism	Section
Protect traffic on the local network	TLS 1.3 with mutual authentication by paired keys	10
Limit what each user can do	Roles, operations, grants, and revocation	11
Know who changed what	The audit log	12
Reach the system from away	VPNs and relays, with end-to-end TLS	13
Act when the network is down	Emergency access over a direct link, limited to the <code>override</code> role	14
Run only genuine firmware	Signed images, verification on the device, A/B images, manifests	15
Keep the owner's data private	Consent, a list of destinations, export, and erasure	16

7 Threat Model

7.1 Assets

In order of importance:

1. **The livestock, and the home.** An attacker who controls equipment can overheat or chill the water, overdose chemicals, stop circulation, or flood the room.
2. **Control.** Equipment that cannot be controlled, or that the owner cannot reach, is a risk even if no one else controls it.
3. **The owner's data and privacy.** History, configuration, schedules, and locations show what the owner keeps, when they are at home, and where they live [OAP-AU Section 21.6].
4. **Credentials and keys**, which give access to everything above.
5. **The home network.** A device that is broken into is a foothold for attacking the owner's other equipment.

7.2 Threats

Table 3 — Threats

#	Threat	In scope	Main protections	Sections
T1	Other people and devices on the home network: guests, housemates, an infected computer, a compromised smart plug. They can watch traffic, connect to any device, and imitate a controller. A malicious web page in a browser on the network can try the same (DNS rebinding).	Yes	No unauthenticated services; mutual TLS with paired keys; host checks on web interfaces.	10
T2	A compromised or malicious app, integration, or service.	Yes	Roles limit what it can do; it can be revoked at once; services can never be owners; the audit log shows what it did.	11, 12
T3	A lost or stolen phone with a paired app.	Yes	Revocation from another app or the controller; owner recovery with physical presence; app locks.	8.4, 11.7
T4	A malicious or compromised cloud service , including a relay or a maker's update server.	Yes	End-to-end encryption through relays; roles for services; signed manifests and images; signing keys kept away from servers.	13, 15
T5	A network attacker on the internet path , for example to an update server.	Yes	Signatures on manifests and images; TLS to internet services.	15
T6	Brief physical access by a visitor, who can read labels and press buttons.	Partly	Setup codes work only in pairing windows; claimed devices need the owner to pair more parties; owners can turn off recovery.	9.3, 11.7
T7	Physical access to the cabinet or the module bus.	Accepted	As with unplugging equipment, someone who can reach the bus can disrupt it. Signed firmware stops them from replacing module firmware.	17

#	Threat	In scope	Main protections	Sections
T8	A stolen, discarded, or second-hand device.	Yes	Secrets are stored encrypted where the hardware allows; factory reset erases owner data and credentials.	8.4, 9.8
T9	A former user or owner: a house-sitter after the stay, a previous owner after a sale.	Yes	Grants expire; revocation; factory reset and re-claiming on transfer.	9.9, 11
T10	Denial of service: Wi-Fi jamming or deauthentication, floods of traffic, repeated pairing attempts, a node jamming the bus.	Partly	It cannot be prevented, only made safe: fail-safe behavior, control that does not depend on the network, rate limits, paths that use no radio link for safety functions.	10.8, 10.9
T11	Malicious firmware, from an attacker or from a rogue insider at a manufacturer.	Partly	Signatures, anti-rollback, the owner's approval, and a record of what was installed. A manufacturer's own signing key is trusted.	15
T12	A compromised module or network device attacking other equipment.	Partly	Network devices accept commands only from their controller. On the module bus, not in this revision (Section 17.2).	11.6, 17

7.3 Out of Scope

This revision does not protect against:

- attacks that need prolonged physical access and laboratory equipment, such as reading chips, fault injection, and side channels;
- a compromised controller, which controls its whole system by design, and the owner's own phones and computers (Section 7.4);
- the owner, who may do anything with their own equipment;
- attacks on the home router or the internet provider, beyond those of an attacker on the network (T1 and T5); and
- the security of mains equipment, which the [Power Module](#) specification addresses (its Section 17).

7.4 Trust Assumptions

- A controller is trusted with everything in its system.
- The owner's apps are trusted with the owner's role.
- A device trusts the parties in its pairing list, each for its role.
- Anyone with physical access to the module bus is trusted, in this revision.
- A manufacturer is trusted with the firmware it signs, but receives no data without the owner's consent.

7.5 Security Objectives

- O1.** Only parties the owner has authorized can see or control a system, and each can do only what its role permits.
- O2.** Every connection on the network is authenticated at both ends, and encrypted and integrity-protected.
- O3.** No attack on the mechanisms of this Specification leaves equipment less safe than its fail-safe behavior.
- O4.** The owner can always regain control of their equipment locally, with physical presence.
- O5.** A device runs only firmware its manufacturer signed, or firmware the owner chose with physical presence.
- O6.** No data leaves the owner's network without the owner's consent, except as Table 12 allows.

8 Identity and Keys

8.1 Identity Keys

- Every device on a network, every controller, and every app installation **MUST** have its own identity key.

2. An identity key MUST be an ECDSA key pair on the P-256 curve [FIPS 186-5].
3. A party MUST generate its identity key itself, from a cryptographically secure random number generator, or have it generated during manufacture by a process that never stores or reuses the private key elsewhere.
4. The private key MUST NOT leave the party: no interface, export, backup, or debug output may reveal it.
5. Modules on the module bus need no identity key in this revision (Section 17).

8.2 Identity Certificates

1. A party presents its identity key in TLS as a self-signed X.509 version 3 certificate [RFC 5280].
2. Its `notAfter` SHOULD be `99991231235959Z`, the value RFC 5280 gives for a certificate with no expiry.
3. Parties MUST NOT use a certificate's validity dates, names, issuer, or extensions to decide whether to trust it. They trust the public key alone (Section 10.3).
4. Parties MAY use raw public keys [RFC 7250] instead of certificates when both support them.

8.3 Key IDs

1. A party's key ID is the SHA-256 hash [FIPS 180-4] of the DER encoding of its public key's `SubjectPublicKeyInfo` [RFC 5280].
2. In JSON and in text, a key ID is written in unpadding base64url [RFC 4648 Section 5], which is 43 characters.
3. When a key ID is shown to people for comparison, it is the first 20 characters of the text form, in groups of four, for example `q8mB 0x4y Zp2V t7Lr Kc3s`.

8.4 Storing Secrets

1. Devices and controllers MUST store private keys, credentials, Wi-Fi passwords, and pairing verifiers where no network or wireless interface can read them.
2. They SHOULD encrypt them at rest with a key held in hardware, such as flash encryption with eFuse keys, and SHOULD disable debug interfaces that could read them in production units (Section 15.8 describes the owner's exception).
3. Apps MUST store identity keys in the platform's protected key store where one exists, and SHOULD protect their use with the platform's screen lock.

8.5 Identity and Privacy

1. A device or controller SHOULD generate a new identity key when it is factory reset (Section 9.8), so that its key cannot be used to follow it from one owner to the next.
2. A controller MUST NOT refuse a device because it cannot prove who manufactured it. This Specification defines no manufacturer attestation, so that equipment owners build themselves works like any other.

9 Setup and Pairing

9.1 No Default Credentials

1. A device, controller, or app MUST NOT ship with a password, key, code, or account that is shared with any other unit, or that can be derived from public information such as a serial number or a network address [ETSI 303 645 provision 5.1-1] [PSTI].
2. It MUST NOT have any credential, account, or interface that lets its manufacturer, or anyone else, in over a network or wireless link without the owner's grant. Wired service connections need physical presence (Section 14.2.3).
3. Until it is claimed, a device or controller MUST accept nothing over a network or wireless link except pairing, and the discovery that pairing needs.

9.2 Setup Codes

1. Every device on a network and every controller MUST come with a setup code of at least eight decimal digits, chosen at random for each unit, which gives about 26.5 bits. Codes made of one repeated digit, and 12345678 and 87654321, MUST NOT be used.
2. The setup code MUST be printed on the unit and on its packaging, as digits and as a QR code [OAP-DP Section 10.4]. A unit that is too small for a label MUST have it on its packaging and in its documentation.
3. A unit with a display SHOULD instead show a new random setup code each time it opens a pairing window.
4. The setup code MUST NOT be usable for anything except claiming the unit, receiving new network settings after a button press (Section 9.6.4), and, on a controller, owner recovery (Section 11.7), each within a pairing window (Section 9.3). It MUST NOT be stored on the unit in a form from which it can be read back. A unit stores the SPAKE2+ verifier made from the code (Section 9.4).

9.3 Pairing Windows

1. An unclaimed unit opens a pairing window when it starts, and with physical presence (Section 9.10), such as a press of its pairing button.
2. A claimed unit opens a pairing window only:
 - for an invitation: a one-time code created over a secure channel, by an owner for another party or by a user for themselves, together with the role that pairing with it grants (Section 11.5; operations `manage-users` and `manage-own-credentials` on a controller, and `open-pairing-window` on a device);
 - for owner recovery on a controller, with its setup code, after physical presence (Section 11.7); or
 - for new network settings, with its setup code, after a button press (Section 9.6.4).

An emergency code needs no pairing window (Section 14.4.2).

3. A pairing window MUST close after a pairing succeeds, or after 20 failed attempts, whichever comes first. A window for a setup code MUST also close 15 minutes after it opens (provisional), and a window for an invitation when the invitation expires (Section 11.5.3). After 20 failed attempts, a new window opens only with physical presence or, for a claimed unit, with a new invitation.
4. A unit MUST NOT run more than one pairing exchange at a time.
5. A unit with an indicator SHOULD show while a pairing window is open.

9.4 Pairing Protocol

Pairing uses SPAKE2+ [RFC 9383], a password-authenticated key exchange. Each attempt lets an attacker test one guess of the code, and the exchange reveals nothing that would allow guesses offline.

1. The party that holds a verifier, such as a device being claimed or a controller issuing an invitation, is the SPAKE2+ Verifier. The party that knows the code, such as the owner's app or a controller adding a device, is the Prover.
2. The ciphersuite is P-256 with SHA-256, HKDF-SHA256, and HMAC-SHA256 [RFC 9383 Section 4].
3. `w0` and `w1` are derived from the code with PBKDF2-HMAC-SHA256 [RFC 8018]:
 - the password is the code's digits or characters, as ASCII, without spaces;
 - the salt is 16 to 32 random bytes, chosen with the verifier;
 - the iteration count is chosen with the verifier, from 1000 to 100 000; Provers MUST support the whole range;
 - the output is 80 bytes: the first 40 are `w0s` and the next 40 are `w1s`, and `w0` and `w1` are each reduced modulo the order of the curve, as RFC 9383 Section 3.2 describes.

The identities are not part of this derivation, so that the verifier can be made when the code is printed. The Verifier sends the salt and iteration count to the Prover before the exchange.

4. The Verifier stores `w0` and $L = w1 \cdot P$ only.
5. When pairing runs inside a TLS connection (Section 10), the transcript inputs are:
 - **Context:** the ASCII string `OAP pairing 1` followed by the 32-byte `tls-exporter` channel binding of the connection [RFC 9266];

- **idProver**: the 32-byte key ID of the Prover's identity key, from its TLS certificate; and
- **idVerifier**: the 32-byte key ID of the Verifier's identity key, from its TLS certificate.

A successful exchange therefore proves that both parties know the code, that the TLS connection has no one in the middle, and which identity key each party holds.

6. When pairing runs over a link without TLS, such as Bluetooth during setup, the binding **MUST** exchange both identity keys before the exchange, **MUST** use their key IDs as **idProver** and **idVerifier**, **MUST** use the **Context OAP pairing 1** followed by a binding-specific label, and **MUST** protect everything sent after the exchange with keys derived from the SPAKE2+ shared secret with HKDF [RFC 5869].
7. When the exchange succeeds, each party records the other's identity key: the Verifier in its user list or pairing list (Sections 11.1 and 11.6), with the role the code grants, and the Prover as a trusted peer. When it fails, the Verifier counts a failed attempt (Section 9.3).

The Device Protocol specification defines how the SPAKE2+ messages are carried [OAP-DP Section 10.2].

9.5 Claiming

1. Claiming is the first pairing of an unclaimed unit. The party that claims it gets the **owner** role on it.
2. **A controller** is claimed by the owner's app. The controller creates a user for the owner, with the **owner** role and the app's identity key as its first credential.
3. **A device** is claimed by the owner's app, or by a controller on the owner's behalf, using the device's setup code. A controller **MUST** claim a device only on the request of a user with the **add-device** permission.
4. When a controller claims a device, the device marks the controller as its controller in its pairing list, because the Prover is a controller [OAP-DP Section 10.2.6]. The controller **MUST** then add the identity keys of the system's owners to the device's pairing list with the **owner** role, at least one for each owner, while leaving at least two entries free for a replacement controller and a service (provisional), so that the owners can move the device to another controller without a factory reset (Appendix B.4). It **MUST** keep these entries up to date as owners and their credentials change.
5. When an owner's app claims a device for a system that has a controller, the app **MUST** add the controller to the device's pairing list with the **owner** role, marked as the device's controller, so that the controller can keep the owners' entries up to date as item 4 requires.
6. Claiming a unit **MUST** be recorded in its audit log (**claimed**), if it keeps one; every controller keeps one (Section 12.1).

9.6 Bluetooth During Setup

1. A unit **MAY** use Bluetooth Low Energy for claiming and for receiving its network settings, such as a Wi-Fi network name and password.
2. Pairing over Bluetooth **MUST** use the protocol of Section 9.4, and **MUST NOT** rely on Bluetooth's own pairing for security.
3. Once a unit has been claimed and has joined its network, it **MUST** stop Bluetooth advertising and **MUST NOT** accept Bluetooth connections, except as item 4 allows. Bluetooth is for setup only, and never for emergency access (Section 14.2).
4. So that an owner can move a claimed unit to a new Wi-Fi network without a factory reset, the unit **MAY** accept Bluetooth again for 15 minutes after a button press (provisional), to receive new network settings from a party that pairs with its setup code. That pairing grants nothing else.

9.7 Modules

Modules on the module bus are configured by the controller as the Module Bus specification defines, without pairing [OAP-MB Section 10]. Section 17 describes the security model of the bus, and how pairing could be added.

9.8 Factory Reset

1. A device or controller **MUST** provide a factory reset by physical presence, such as holding a button for 10 s, that no setting can disable. An owner **MAY** also reset a unit over the network (operation **factory-reset**).

2. A factory reset MUST erase:
 - the pairing list or user list, and all credentials;
 - network settings, such as Wi-Fi passwords;
 - the audit log, history, configuration, consent records, and every other piece of owner data; and
 - the identity key, which SHOULD be replaced (Section 8.5).
3. It restores the manufacturer's default settings, including the shipped fail-safe settings [OAP-FS Section 13.1], and leaves the firmware as it is.
4. During and after a factory reset, outputs MUST behave as after a power-up [OAP-FS Section 9.1]. A reset device has no controller, so its outputs take their fail-safe behavior until it is claimed and controlled again.
5. Erasure MUST make the data unrecoverable through any interface of the unit [ETSI 303 645 provision 5.11-1].

9.9 Transferring Ownership

1. An owner gives a system to someone else by adding the new owner (Section 11.5) and then removing themselves, or by factory resetting the equipment.
2. An owner selling or giving away a single device SHOULD factory reset it first. A controller MUST offer to do this when an owner removes a device from the system.

9.10 Physical Presence

1. Where this Specification requires physical presence, a unit MUST require a press of a button on the unit, or the entry of a code that the unit shows on its own display.
2. A unit MUST NOT accept as physical presence anything that can be done over a network, such as removing and restoring its power, which a smart plug can do, or a message from a computer connected to it.
3. Starting up opens a pairing window only on an unclaimed unit (Section 9.3), which has nothing yet to protect.

10 Secure Channels on the Local Network

10.1 TLS 1.3

1. Every connection between parties over an IP network MUST be protected with TLS 1.3 [RFC 9846], with both parties authenticated by their identity keys, except:
 - multicast DNS and DNS service discovery [RFC 6762] [RFC 6763];
 - a plain HTTP listener that does nothing but redirect to HTTPS; and
 - the time and network services of Table 12.
2. TLS 1.2 and earlier versions MUST NOT be used.
3. This holds on every network, including for battery-powered devices on Thread [OAP-DP Section 14.3] (Appendix D, issue 1).

10.2 TLS Profile

1. Parties MUST support the cipher suite `TLS_AES_128_GCM_SHA256`, the key exchange group `secp256r1`, and the signature scheme `ecdsa_secp256r1_sha256`. They SHOULD also support `x25519` and `TLS_CHACHA20_POLY1305_SHA256`.
2. A party accepting a connection MUST request the other party's certificate, and the other party MUST send its identity certificate. The exception is a connection from a web browser to a web interface (Section 10.4), which MUST NOT request a certificate, because browsers authenticate with passwords.
3. Parties MUST NOT send or accept early data (0-RTT), because it can be replayed, and a replayed command can run equipment twice.
4. Parties MAY resume sessions, but only with `psk_dhe_ke`, and MUST NOT resume a session whose credential has been revoked or whose grant has expired.

5. Parties SHOULD support the record size limit extension [RFC 8449], so that small devices need less memory.

10.3 Authenticating the Other Party

1. Each party MUST check that the identity key in the other party's certificate is one it trusts:
 - a controller trusts the credentials in its user list (Section 11.1) and the devices it has paired;
 - a device trusts the keys in its pairing list (Section 11.6); and
 - an app trusts the controllers and devices it has paired with.
2. A party MUST refuse every request on a connection from a key it does not trust, except pairing (Section 9.4) and password sign-in (Section 10.5).
3. A valid message from a device's controller [OAP-FS Section 7.1] is one received on a secure channel from the controller's identity key. A device MUST NOT accept supervision messages, or endpoint commands other than `identify`, from any other key, and accepts attribute writes from other keys only as Table 6 permits their role. Messages from other keys never count as contact.
4. Only a party marked as a controller in a device's pairing list, with the `owner` or `full-control` role, may take control of the device, and only as Fail-Safe Behavior Section 7.5 allows (Section 11.6.3). Over the Device Protocol, it takes control with `control.take` [OAP-DP Section 12.1]; `command-endpoint` is what the party in control may then do. When a device gets a new controller, each of its output and unit endpoints that is not yet in its fail-safe behavior enters it at once, as Fail-Safe Behavior Section 7.5.3 requires, so that no endpoint goes on with what the previous controller commanded.

10.4 Web Interfaces

A browser cannot check a paired key, so it warns about a self-signed certificate. An attacker on the local network who intercepts the first connection can then pose as the device. An app that checks the unit's key, and relays the web interface to the browser, avoids this (item 6; Appendix D, issue 4).

1. A device or controller MAY offer a web interface. It MUST serve it over HTTPS with its identity key. It SHOULD present a certificate for that key that browsers accept: with the unit's names in a subject alternative name, the `serverAuth` extended key usage, and a validity of no more than 398 days, renewed automatically, because some browsers refuse certificates that last longer (provisional; not verified on current browsers). Paired apps, and the controller's display if it has one, SHOULD show the SHA-256 fingerprint of that certificate, so that the owner can compare it with the one the browser shows. A fingerprint shown on the web page itself proves nothing, because an attacker would serve the page.
2. The web interface MUST reject any request whose `Host` header is not one of the unit's own names or addresses, which stops DNS rebinding.
3. It MUST NOT allow requests from other origins (CORS).
4. Requests that change anything MUST be protected against cross-site request forgery, for example with `SameSite=Strict` cookies and a check of the `Origin` header.
5. Sessions MUST use cookies marked `Secure`, `HttpOnly`, and `SameSite=Strict`, and MUST end after 30 minutes without use and after 12 hours in all (provisional).
6. Owners SHOULD reach a web interface through a paired app, which checks the unit's key, rather than directly. An app MAY serve a unit's web interface to a browser on the same phone or computer: it listens on the loopback address, and relays each of the browser's requests to the unit over a connection that it authenticates with its identity key, as the Device Protocol defines [OAP-DP Section 15.5], so that the browser never connects to the unit, and the unit serves the app's user without a password. Any program on the same phone or computer can reach the loopback address, so an app that relays a web interface MUST:
 - listen on the loopback address only;
 - accept requests only with a secret that it chooses at random each time it opens the interface for a person, and gives only to the browser that it opens;
 - reject any request whose `Host` header is not the loopback address and port it listens on, which stops DNS rebinding;

- protect requests that change anything against cross-site request forgery, as item 4 requires of a unit; and
- stop relaying when the person closes the interface, and when its credential is revoked (Section 10.6).

An owner MAY turn the web interface off (operation `change-network-settings`).

10.5 Passwords

Passwords are for web browsers and for services that cannot hold an identity key. Apps use identity keys.

1. A password MUST be chosen by its user. It MUST be at least 15 characters long, and MUST be checked against a list of common and compromised passwords [NIST 800-63B] (provisional). Lengths up to at least 64 characters MUST be allowed, and the unit MUST NOT impose any other rule on its composition.
2. A unit MUST store only a salted hash of the password, with PBKDF2-HMAC-SHA256 [RFC 8018] of at least 100 000 iterations, or a memory-hard function such as scrypt [RFC 7914] or Argon2id [RFC 9106] (provisional).
3. A password MUST be sent only inside a TLS connection.
4. A unit MUST limit failed sign-ins, so that no more than 100 guesses can be made for one user in 24 hours (provisional), and MUST record failures in the audit log (`sign-in-failed`).

10.6 Revocation

1. When a credential is revoked or a grant expires, every party that trusted it MUST close each connection authenticated with it within 5 s (provisional), and MUST refuse new ones.
2. When an entry is removed from a device's pairing list, the device MUST do the same for that key. If the removed key was its controller's, the device has lost contact [OAP-FS Section 7].

10.7 Time

1. Authentication MUST NOT depend on the clock. An invalid clock [OAP-FS Section 10.5] never stops a party from authenticating its controller, its devices, or its users.
2. While its clock is not valid, a controller cannot tell whether a grant with an expiry has ended. It MUST then limit users with such grants to `view-state`, `stop-all`, and overrides that stop an output, so that a house-sitter can still stop equipment after a power cut, and MUST record the clock's state in its audit log (`clock-invalid`). Owners, and users whose grants have no expiry, are unaffected.
3. A controller SHOULD use authenticated time, such as Network Time Security [RFC 8915], where its time source offers it. When its battery-backed clock is valid, a controller SHOULD NOT move its clock by more than 1 hour on the word of an unauthenticated time source without asking an owner (provisional), because the clock drives schedules, programs, and the expiry of grants.

10.8 Network Exposure

1. A unit MUST NOT listen on any port, or offer any service, that the Device Protocol specification [OAP-DP] or this Specification does not need [ETSI 303 645 provision 5.6-1].
2. Production firmware MUST NOT include debugging or update services that bypass this Specification, such as unauthenticated remote consoles or firmware upload ports.
3. A unit MUST NOT open ports on the owner's router, for example with UPnP, NAT-PMP, or PCP.
4. Before a party is authenticated, a unit MUST reveal no more than discovery needs. Discovery records SHOULD NOT contain names the owner has set, the owner's name, or serial numbers.

10.9 Resisting Denial of Service

1. A controller MUST keep running its control functions, and supervising its devices, while its network interface is flooded with traffic or connection attempts. Control MUST NOT wait on the network.
2. A unit MUST limit the number of connections it accepts, and MUST keep at least one free for a party that is already paired.
3. A unit SHOULD support WPA3-Personal and Protected Management Frames, which resist forged disconnections, and MUST support WPA2-Personal. It MUST NOT require an open Wi-Fi network.

- Every safety function needs a path that uses no radio link: wired to its controller, or protection inside the device, as Fail-Safe Behavior Section 13.4 defines safety functions and such paths [OAP-FS Sections 5.2 and 13.4] [P0001]. A controller MUST warn its owners, persistently, while a safety function has no such path, and MUST NOT refuse the configuration for that reason [OAP-FS Sections 13.4.6 and 13.4.7]. Protecting a radio link does not make it such a path, because jamming and forged disconnections can still interrupt it (Table 3, T10).

11 Users, Roles, and Sharing

11.1 Users

A controller keeps a list of users.

Table 4 — Members of a user

Member	Description
<code>id</code>	A UUID [RFC 9562] that identifies the user on this controller. It never changes, and it is not reused.
<code>name</code>	A name for people, such as “Alex” or “Home automation”.
<code>kind</code>	<code>person</code> or <code>service</code> .
<code>role</code>	<code>view</code> , <code>override</code> , <code>full-control</code> , or <code>owner</code> (Section 11.2).
<code>grantExpires</code>	When the grant ends. Optional. The Device Protocol carries it with this name [OAP-DP Section 10.5].
<code>remote</code>	Whether the user may connect remotely (Section 13.1).
<code>credentials</code>	The user’s credentials: the key IDs of app installations, each with a label such as “Alex’s phone” and the time it was added; a password hash; or an emergency code verifier.
<code>created</code> , <code>createdBy</code>	When, and by which user, the user was created.

- A controller MUST support at least 16 users and 32 credentials (provisional).
- Another Open Aquarium specification that records who did something, such as the `by` member of an override [OAP-AU Section 15], uses the user’s `id`.
- A service MUST NOT have the `owner` role, and MUST NOT use emergency access.
- A controller MUST always have at least one owner. It MUST refuse to remove or demote its last owner, except by factory reset.

11.2 Roles

Each role includes everything the one before it allows. Table 6 gives the operations that each permits (Appendix D, issue 13).

Table 5 — Roles

Role	Name	Includes	Adds
<code>view</code>	View	—	See the system: readings, states, faults, alarms, history, and configuration.
<code>override</code>	Override	<code>view</code>	Also operate the system for a while: overrides, modes, alarms, feeding, tests, and stopping equipment. It cannot add chemicals to the water or change how the system works.
<code>full-control</code>	Full control	<code>override</code>	Also change how the system works: automation, device settings, fail-safe settings, devices, and firmware.
<code>owner</code>	Owner	<code>full-control</code>	Also decide who has access and what leaves the owner’s network: users, sharing, remote access, emergency access, privacy, and ownership.

11.3 Operations

Table 6 gives, for each operation, the smallest role that permits it on a controller and on a device. It is also published in machine-readable form in `permissions.json`. A dash means that the operation does not apply there. `controller` means that a device accepts the operation only from the party in control of it (Section 11.6).

Table 6 — Operations and the roles that permit them

Group	Operation	Controller	Device	Audited	Description
View	view-state	view	view	—	See current readings, output states, faults, alarms, active modes, and overrides.
View	view-history	view	view	—	See recorded history: observations, events, and output states.
View	view-configuration	view	view	—	See the automation configuration, device settings, fail-safe settings, and firmware information.
View	identify	view	view	—	Make a device or endpoint show where it is, with the <code>identify</code> command. It changes no output.
View	manage-own-credentials	view	—	Yes	Add or remove one's own credentials, for example a new phone, and sign out other sessions. Users cannot change their own role.
Operate	acknowledge-alarm	override	—	Yes	Acknowledge or silence an alarm.
Operate	set-override	override	—	Yes	Set or end an override of an output (Automation Section 15).
Operate	start-mode	override	—	Yes	Start a mode of any kind, or end one other than an emergency mode (Automation Section 14), and start or end a scene of a lighting function (Lighting Programs Section 13).
Operate	end-emergency-mode	override	—	Yes	End an emergency mode after checking the system (Automation Section 14.7).
Operate	clear-latch	override	controller	Yes	Clear a latched condition or a latched fault (Automation Section 10.4; <code>resetFault</code>).
Operate	stop-all	override	controller	Yes	Stop every output at once, and keep them stopped until it expires or is ended (Section 11.8).
Operate	run-action	override	controller	Yes	Run a one-off action that adds no chemicals and that a device limits by itself: feed now, or start a test.
Operate	record-maintenance	override	controller	Yes	Record maintenance that changes no output, such as a refilled reservoir or container, a replaced reagent, or an emptied waste container.
Configure	dose-now	full-control	controller	Yes	Run a manual dose, or prime or reverse a dosing head, within the device's limits.
Configure	enter-results	full-control	—	Yes	Enter test results, or import laboratory documents, which can change how much a dosing function adds, so that an owner grants it on purpose.
Configure	start-program	full-control	—	Yes	Start or stop a step program (Automation Section 16).
Configure	change-configuration	full-control	—	Yes	Change or import the automation configuration (Automation Sections 7 and 19).
Configure	change-device-settings	full-control	full-control	Yes	Change the config attributes of a device or endpoint, including names and safety limits, other than fail-safe settings.

Group	Operation	Controller	Device	Audited	Description
Configure	change-fail-safe-settings	full-control	full-control	Yes	Change <code>failSafeAction</code> , <code>failSafeState</code> , <code>failSafeTimeout</code> , <code>powerOnBehavior</code> , <code>powerOnDelay</code> , <code>loadCategory</code> , or <code>backupThermostat</code> , the settings of a backup thermostat (Fail-Safe Behavior Section 13; Device Classes Section 9.1).
Configure	calibrate	full-control	controller	Yes	Calibrate a sensor or a dosing head.
Configure	command-endpoint	—	controller	—	Command an endpoint, and send supervision messages, as its controller (Fail-Safe Behavior Section 7.5). Users command endpoints only through the controller's operations.
Configure	add-device	full-control	—	Yes	Add a device to the system by pairing it with the controller, or accept a new module.
Configure	remove-device	full-control	—	Yes	Remove a device from the system. The controller releases it and leaves its pairing list.
Configure	restart	full-control	full-control	Yes	Restart a device or the controller.
Configure	install-firmware	full-control	full-control	Yes	Install a firmware update, return to the previous firmware, change the update channel, or turn automatic installation on or off.
Configure	view-audit-log	full-control	full-control	—	See the audit log.
Configure	export-data	full-control	full-control	Yes	Export history, configuration, and the audit log.
Manage	view-access	owner	owner	—	See the users, their roles and credentials, and the pairing lists.
Manage	manage-users	owner	—	Yes	Invite users, change their roles, expiry, and remote access, and revoke users or their credentials; turn on a Matter bridge, open its commissioning window, and remove its fabrics (Section 13.4.4).
Manage	manage-pairings	—	owner	Yes	Add or remove entries in a device's pairing list, change their roles, or mark which may act as its controller.
Manage	open-pairing-window	—	owner	Yes	Create an invitation for another party on a device that has already been claimed: a one-time code, and the role that pairing with it grants (Section 11.5). On a controller, <code>manage-users</code> invites a new user, and <code>manage-own-credentials</code> a user's own new app.
Manage	configure-remote-access	owner	—	Yes	Turn remote access on or off, and choose its relay or other path.
Manage	configure-emergency-access	owner	—	Yes	Turn emergency access on or off, and set or remove emergency codes.
Manage	change-network-settings	owner	owner	Yes	Change Wi-Fi networks, the controller's own access point, and other network settings.
Manage	manage-consent	owner	owner	Yes	Give or withdraw consent to send data off the owner's network: update checks, telemetry, and cloud services.

Group	Operation	Controller	Device	Audited	Description
Manage	erase-data	owner	owner	Yes	Erase history and other personal data without a factory reset (Section 16.6).
Manage	allow-owner-firmware	owner	owner	Yes	Allow a device to run firmware that its manufacturer did not sign. Also needs physical presence.
Manage	factory-reset	owner	owner	Yes	Erase a device's or the controller's owner data, credentials, and settings. Physical presence also allows it.

1. A controller MUST check the role of the user behind every request before carrying it out, and MUST refuse operations the role does not permit.
2. A device MUST do the same for the parties in its pairing list.
3. When a controller carries out an operation on a device for a user, the controller is responsible for checking the user's role; the device checks the controller's.
4. Where another Open Aquarium specification says that the owner does or confirms something, a user whose role permits the matching operation in Table 6 does it. For example, the owner's confirmation of a fail-safe change [OAP-FS Section 13.1] is given by a user with the `full-control` role, at the time of the change, and ending an emergency mode [OAP-AU Section 14.7] needs the `override` role.
5. Stopping equipment needs no more than the `override` role (`stop-all`, `set-override`). Nothing in this Specification stops a controller from putting an endpoint in its inactive state [OAP-FS Section 8.2].
6. A controller SHOULD ask a user to confirm again, or to authenticate again, before `factory-reset`, `manage-users`, `allow-owner-firmware`, and `install-firmware`.

11.4 Grants and Expiry

1. An owner MAY give any user other than an owner an expiry, for example a house-sitter's grant that ends when the owner returns.
2. When a grant expires, the controller MUST treat the user as revoked (Section 10.6), MUST record `grant-expired`, and SHOULD keep the user, without access, so that the owner can renew it.
3. Overrides and modes that a user started do not end because the user's grant ends; they end as the Automation specification defines.

11.5 Sharing

1. An owner shares access by creating an invitation for a new user, with a role, an expiry, and whether remote access is allowed (operation `manage-users`).
2. A user adds a new app installation by creating an invitation for themselves from an app that is already paired (operation `manage-own-credentials`). The invitation adds a credential to the same user and grants nothing new.
3. An invitation is a one-time code of at least eight decimal digits, chosen at random, from which the controller makes a verifier (Section 9.4). An owner can also create an invitation on a device (operation `open-pairing-window`), for example for a second controller or a home automation service; the owner's app gives the device the verifier and the role over a secure channel. It MUST expire after 24 hours unless the owner chooses a shorter or longer time, of at most 7 days (provisional), and it MUST be usable only once.
4. The invited user's app pairs with the controller using the invitation, on the local network or through a relay (Section 13).
5. A controller MUST show its owners each new credential, with the user's name and the app's label, and MUST record it (`paired`).

11.6 Device Pairing Lists

A device that is not a controller keeps a pairing list instead of a user list.

1. Each entry holds a key ID, a role, a label, when it was added, and whether the party may act as the device's controller. Entries do not expire, so that a device never loses its controller because of its clock.

2. A device MUST support at least 16 entries (provisional).
3. Only an entry marked as a controller, and with the `owner` or `full-control` role, may take control of the device (Fail-Safe Behavior Section 7.5; Section 10.3.4). Operations whose device column in Table 6 says `controller` are accepted only from the party in control. An owner's app has the `owner` role but is not marked as a controller, so it can manage the device but cannot take it over from the controller.
4. A typical device lists its controller, marked as a controller, and the system's owners, all with the `owner` role (Section 9.5). A home automation service that reads the device directly has the `view` role.
5. Only parties with the `owner` role may change the list (operation `manage-pairings`). A device MUST NOT remove its last `owner` entry, except by factory reset.

11.7 Revocation and Recovery

1. An owner MAY revoke any user, and any credential of any user (operation `manage-users`). Any user MAY revoke their own credentials (operation `manage-own-credentials`), for example the credential of a lost phone, from another of their app installations.
2. When an owner's credential is revoked, the controller MUST remove its key from the pairing lists of the system's devices.
3. **Owner recovery.** An owner who has lost every credential can regain access with physical presence:
 - physical presence at the controller (Section 9.10) opens a recovery pairing window, during which the controller's setup code pairs a new app as a new user with the `owner` role;
 - the controller MUST record `owner-recovery`, and SHOULD notify every owner, as the Alarms and Notifications specification defines [OAP-AN Section 13.4.1]; and
 - an owner MAY turn recovery off, after which a lost owner can only factory reset the controller.
4. Recovery keeps the configuration and history. A factory reset (Section 9.8) erases them.

11.8 Stopping Everything

The `stop-all` operation is for a person who sees something wrong, such as a leak, and wants every piece of equipment stopped at once.

1. `stop-all` MUST stop every output of the system at once, except each output that an alarm function drives [OAP-AN Section 15], which it leaves to that function, as power-loss mode does [OAP-FS Section 9.5], so that a siren or a strobe keeps signaling critical alarms, `stop-all` itself among them.
2. The controller MUST keep the outputs stopped, at the precedence of interlocks [OAP-AU Section 17.3], until `stop-all` expires or a user with the `override` role ends it. Its expiry is chosen by the user: 1 hour by default, and at most 24 hours (provisional), so that its stops end by themselves within a day even if the controller fails (item 3). A controller MUST refuse a longer expiry.
3. The controller MUST send each output's stop with `revertAfter` for the time left, so that the stops end even if the controller fails [OAP-AU Section 17.5], as follows:
 - the commands that end an operation, the `stop` of a doser, a feeder, or a filter roller, carry no `revertAfter`, and an operation that a stop ends never resumes by itself [OAP-DC Section 7.14];
 - a dosing head that runs its own program is also held, with `set`, `enabled false`, and `revertAfter`, so that its program starts no dose until the hold ends [OAP-DC Section 7.13], and the hold and the `stop` may go in either order, because a command that ends an operation leaves a pending revert as it is [OAP-DC Section 7.4.7];
 - over the Device Protocol, the controller MAY instead send one `device.stop` with `revertAfter` to a whole device none of whose outputs an alarm function drives, which ends operations in the same way [OAP-DP Section 11.4]; and
 - on the module bus, it sends each endpoint's stop at the immediate priority [OAP-MB Section 8.3], and sends `COMMAND_EMERGENCY_STOP` [OAP-MB Section 11.7] to each module that has not accepted all of its stops within 1 s (provisional), and to no other, and never to a module with an output that an alarm function drives, because `COMMAND_EMERGENCY_STOP` stops every output and carries no `revertAfter`, and would cancel the reverts just sent [OAP-DC Section 7.4.7].
4. `stop-all` MUST be available to every user with the `override` role or above, including in emergency sessions and while the clock is not valid (Section 10.7), and on the controller's own buttons (Section 14.2.1).

- When `stop-all` ends, each output returns to the state its control function, modes, and overrides give it. Its stops end no latched stop, so an output that an interlock or an emergency mode holds with a latched stop stays stopped, also when the reverts of `stop-all` fall due on a device whose controller has failed [OAP-FS Section 8.4.3].
- The controller MUST NOT delay the stops of `stop-all`, and MUST send them to endpoints under local control too [OAP-AU Section 17.4.2]; devices accept them at any time [OAP-FS Section 8.2].

12 Audit Log

12.1 What Is Recorded

- A controller MUST record in its audit log every operation that Table 6 marks as audited, whether it succeeds, fails, or is refused, and every event of Table 8.
- Changes to fail-safe settings MUST be recorded with their old and new values, which meets Fail-Safe Behavior Section 13.1.4.
- A device SHOULD keep an audit log of the operations performed on it, and of the events of Table 8 that concern it. A device with little storage MAY record only `claimed`, `paired`, `pairing-window-closed`, `firmware-installed`, `firmware-reverted`, `firmware-refused`, and `manage-pairings`.
- Viewing is not recorded.
- A controller with a Matter bridge MUST record each fabric added to or removed from the bridge, however it was added or removed (`fabric-added`, `fabric-removed`) [OAP-MM Section 7.3]. Any administrator of a fabric can add another fabric through Matter's own commissioning, without the owner, so the record is how owners see who can reach the bridge.
- Every record of what a Matter bridge does, these included, MUST have the bridge's user as `actor`, `via` `matter`, and the fabric in `details`, as `fabricIndex`, its index on the bridge, `label`, and `vendorId` [OAP-MM Sections 7.3 and 12.1].

12.2 Records

Table 7 — Members of an audit record

Member	Presence	Description
<code>seq</code>	REQUIRED	A number that increases by at least 1 with each record. It never goes back, not even after a restart, so a missing record leaves a gap.
<code>time</code>	REQUIRED	When it happened, as an RFC 3339 timestamp [RFC 3339], or <code>null</code> while the clock is not valid.
<code>boot</code> , <code>uptime</code>	OPTIONAL	How many times the unit has started, and the time since it last started, in seconds. REQUIRED when <code>time</code> is <code>null</code> .
<code>actor</code>	REQUIRED	Who acted: its kind (person, service, controller, device, physical, system, or unknown), and, where known, the user's <code>id</code> as <code>user</code> , their <code>name</code> , the <code>credential</code> used, and their <code>role</code> .
<code>via</code>	REQUIRED	How: <code>local</code> , <code>remote</code> , <code>emergency</code> , <code>physical</code> , <code>bus</code> , <code>internal</code> , or <code>matter</code> , for what a Matter bridge does for a fabric (Section 12.1.6).
<code>action</code>	REQUIRED	The operation (Table 6) or event (Table 8), or an extension name.
<code>target</code>	OPTIONAL	What was acted on: a device in the form of Automation Section 8.5, an <code>endpoint</code> , an automation element, a user, or a <code>credential</code> .
<code>result</code>	REQUIRED	<code>success</code> , <code>denied</code> , or <code>failed</code> .
<code>details</code>	OPTIONAL	Anything more, such as old and new values, or a count of repeated failures.

For example:

```
{ "seq": 3, "time": "2026-10-10T09:24:10+10:00", "boot": 1, "uptime": 1038.2,
  "actor": { "kind": "person", "user": "3f6c2a1e-8b4d-4f7a-9c2e-5d1b7a9e0c34", "name": "Alex", "role": "owner" },
  "via": "local", "action": "change-fail-safe-settings",
  "target": { "device": "device-protocol:Rj4mT9wK2bX7nP1qV6sZ0cH5yD8fL3gA4eU9i02kN7s", "endpoint": "head_1" },
  "details": { "changes": [{ "attribute": "failSafeTimeout", "from": 120, "to": 60 } ] }, "result": "success" }
```

Table 8 — Audit events

Event	Meaning
<code>claimed</code>	The device or controller was claimed by its first owner.
<code>paired</code>	A credential was added by pairing: a user's app, a controller, or an invited user.
<code>pairing-failed</code>	A pairing attempt failed. Repeated failures MAY be recorded as one record with a count.
<code>pairing-window-closed</code>	A pairing window closed because too many attempts failed.
<code>signed-in</code>	A user signed in with a password.
<code>sign-in-failed</code>	A sign-in failed. Repeated failures MAY be recorded as one record with a count.
<code>grant-expired</code>	A user's time-limited grant ended.
<code>invitation-expired</code>	An invitation ended unused.
<code>owner-recovery</code>	An owner credential was added by recovery with physical presence (Section 11.7).
<code>remote-session</code>	A remote session started or ended.
<code>emergency-session</code>	An emergency access session started or ended.
<code>fabric-added</code>	A Matter fabric was added to the controller's Matter bridge, however it was added (Matter Mapping Section 7.3).
<code>fabric-removed</code>	A Matter fabric was removed from the controller's Matter bridge, however it was removed (Matter Mapping Section 7.3).
<code>firmware-installed</code>	New firmware was installed and confirmed.
<code>firmware-reverted</code>	New firmware failed to confirm, and the device returned to its previous firmware.
<code>firmware-refused</code>	A firmware image or update manifest failed verification.
<code>clock-invalid</code>	The clock became not valid, or valid again (Fail-Safe Behavior Section 10.5).
<code>node-joined</code>	A node joined the module bus for the first time.
<code>node-conflict</code>	Two nodes on the module bus used the same node-ID, or a node other than the controller acted as a controller.
<code>recorded-gap</code>	Records were lost, for example because storage failed; <code>details</code> gives how many.

12.3 Keeping and Exporting the Log

1. A controller MUST keep at least the latest 2000 records, or the records of the last 90 days if they are more, in storage that survives power loss (provisional). When it must drop records, it drops the oldest.
2. Records MUST NOT be changed, and MUST NOT be deleted except by dropping the oldest or by factory reset.
3. Users with the `full-control` role or above can view the log, and export it as a JSON document valid against `schema/audit-log.schema.json` [JSON Schema] (operations `view-audit-log` and `export-data`).
4. Records contain names the owner chose, and so are the owner's data (Section 16). They MUST NOT contain passwords, codes, or private keys.

13 Remote Access

13.1 Requirements

1. Remote access MUST NOT be needed for any function. Everything a user can do remotely MUST also be possible on the local network.
2. Remote access MUST be off until an owner turns it on (operation `configure-remote-access`). An owner MUST be able to turn it off on the local network at any time, and it MUST then stop within 5 s (provisional).
3. Remote connections MUST use the same secure channels (Section 10), credentials, and roles as local ones, with TLS from end to end between the app and the controller. A relay or other intermediary MUST NOT hold a key that lets it read or change the traffic.
4. A controller MUST refuse remote connections from users whose grants do not allow remote access. Grants for owners allow it by default; other grants do not, unless the owner chooses (provisional).
5. A controller MUST connect only outwards to reach a relay, and MUST NOT accept connections from the internet otherwise (Section 10.8).

6. Devices other than the controller SHOULD NOT be reachable remotely except through the controller.
7. Losing remote access MUST NOT affect control.
8. A controller MUST record each remote session (`remote-session`).

13.2 Architectures

Table 9 — Ways to reach a system remotely

Architecture	How it works	Needs from the controller	Who sees what	Status
A. VPN to the home network	The owner's router or home server runs a VPN, such as WireGuard, or the owner uses an overlay network. The app joins the home network, and connects as if local.	Nothing.	Only the owner's own equipment, and an overlay network's coordination service, if one is used, which sees which devices connect.	Works with any conforming controller. RECOMMENDED for owners who already run a VPN.
B. Owner-run relay	The controller keeps an outbound connection to a relay that the owner runs on a home server or a rented server. Apps connect to the relay, which joins the two connections.	The relay protocol.	The relay's host sees when and how much the owner connects, and from where. It cannot read or change anything.	The default (Appendix D, issue 3).
C. Relay service	As B, but the relay is run by someone else: a manufacturer, the community, or a business.	The relay protocol.	As B, for the service's operator.	Optional. The owner chooses the service, and can change it.
D. Port forwarding	The owner opens a port on the router to the controller.	Nothing more.	Anyone on the internet can reach the controller's interface.	NOT RECOMMENDED. Controllers MUST NOT set it up themselves.

The relay protocol of architectures B and C is one open protocol, so that owners can switch between running their own relay and using a service, and any app works with any relay (Appendix D, issue 9).

13.3 Relays

1. A relay MUST forward the TLS connection between app and controller without ending it.
2. A controller MUST authenticate to a relay with its identity key, so that only controllers the relay serves can use it, and SHOULD NOT need to give the relay anything else, such as an email address.
3. An app MUST reach a controller through a relay by its key ID, and MUST check the controller's identity key as on the local network (Section 10.3).
4. Invitations MAY be accepted through a relay (Section 11.5).
5. Notifications that pass through a service SHOULD be encrypted from end to end, for example as Message Encryption for Web Push does [RFC 8291], as the Alarms and Notifications specification defines [OAP-AN Section 17.2.3].

13.4 Cloud Services and Integrations

1. A cloud service, a home automation server, or another integration is a user of kind `service`, with a role an owner grants. It MUST NOT have the `owner` role (Section 11.1).
2. An owner MUST be able to see and revoke every service's access on the local network, and the service MUST NOT be able to prevent it.
3. Local and cloud access work at the same time: connecting a service MUST NOT turn off local access or local integrations.
4. A Matter bridge gives Matter controllers access as the Matter Mapping specification defines [OAP-MM]. Each bridge is one user of kind `service`, whose role caps what every Matter fabric can do, whatever Matter's access control grants; fabrics are not users, and have no roles of their own (Appendix D, issue 23). A bridge MUST NOT let a Matter fabric do more than the role an owner granted the bridge's user. Turning a bridge on creates its user, so it needs `manage-users`, and, because a home ecosystem may send what the bridge presents off the owner's network, `manage-consent` [OAP-MM Section 7.2]; it has no operation of its

own. Opening the bridge's commissioning window, and removing a fabric from it, also need `manage-users`, because each gives or takes away access.

- Administrators of a Matter fabric MAY add other fabrics to the bridge through Matter's own commissioning windows [OAP-MM Section 7.3]. The controller MUST record each fabric added or removed (Section 12.1.5), and MUST notify every owner of each, as Alarms and Notifications Section 13.4.4 requires (`fabric-added`, `fabric-removed`) [OAP-AN].

14 Emergency Local Access

14.1 Purpose

When the router fails or the home network is down, an owner, or a neighbor who has been given a code, may need to stop a heater or run a pump. Emergency access gives them a direct link to the controller for overrides only.

14.2 Paths

- A controller with its own buttons or display MUST let a person there stop all outputs (`stop-all`), with no credential.
- A controller SHOULD offer emergency access over a direct wireless link: its own Wi-Fi access point (Section 14.6). Bluetooth is for setup only, and is never used for emergency access (Section 9.6.3).
- A wired service connection on the controller, such as a USB console, MUST NOT be reachable from any network. It does not count as physical presence by itself, because a computer may be left connected to it: operations that need physical presence also need a button press on the unit.

14.3 When It Is Available

- Emergency access over a wireless link MUST be off until an owner turns it on (operation `configure-emergency-access`).
- Once turned on, the link MUST be available when, and only when:
 - after the controller has had no working connection to the home network for 2 minutes (provisional); or
 - for 30 minutes after physical presence at the controller (provisional).
- The link MUST close 30 minutes after it was last used (provisional), or when the home network returns and no emergency session is open.

14.4 Credentials

- The key credentials of any user with the `override` role or above work for emergency access, with the same secure channel and pairing protocol as on the local network (Sections 9 and 10).
- An owner MAY create emergency codes for people without a paired app, such as a neighbor. An emergency code has at least eight random digits, and is a credential of a user of kind `person` with the `override` role that works only for emergency access. Each session with a code begins with the exchange of Section 9.4, with the code in place of a setup code, and the controller does not keep the phone's key afterwards. An emergency code needs no pairing window: the controller accepts it only over its emergency link while the link is available (Section 14.3), and MUST refuse it at any other time, and during the lockout of item 3 [OAP-DP Section 10.2.1]. Such a user MUST NOT be able to add other credentials.
- After 10 failed attempts with emergency codes, the controller MUST refuse emergency codes until physical presence at the controller, or an owner, allows them again (provisional).

14.5 Limits

- An emergency session MUST be limited to the operations that Table 6 permits the `override` role, other than `manage-own-credentials`, whatever the user's role. Every session over the emergency link (Section 14.6) is an emergency session, whatever credential opened it, and the controller tells the app so [OAP-DP Section 9.4]. Configuration, users, credentials, firmware, fail-safe settings, and network settings cannot be changed in an emergency session.

2. Overrides set in an emergency session expire as Automation Section 15 requires, and interlocks still apply.
3. The controller MUST record the start and end of each emergency session (`emergency-session`), and each operation in it with `via` set to `emergency`.

14.6 The Direct Link

1. A controller's emergency access point MUST use WPA2-Personal or WPA3-Personal, with a passphrase unique to the unit. Paired apps receive the passphrase over a secure channel; it MAY also be printed on the controller for people with only an emergency code. A controller that hosts a device network [P0001], and can host only one access point, MAY instead offer emergency access on its device network (Appendix D, issue 34), but only if:
 - it isolates the device network's clients from one another, so that a party that joins the network reaches only the controller;
 - on the device network, it accepts emergency sessions only while Section 14.3 makes the link available, and no other session from a party that is not one of its devices; and
 - it gives the device network's passphrase to people only as this item allows for the access point's.
2. The access point MUST NOT route traffic to any other network, and parties that join it, or a device network, for emergency access MUST be able to reach nothing but emergency access on the controller.

15 Firmware Updates

15.1 Signed Images

1. Every firmware image that a device or controller runs MUST be signed by its manufacturer, or by its owner as Section 15.8 allows. In this section, "device" includes controllers and modules.
2. Signatures MUST give at least 128-bit security, for example ECDSA with P-256 and SHA-256, Ed25519, or RSA-PSS with a 3072-bit key.
3. The format of images and their signatures is the manufacturer's choice, because it is tied to the device's boot code. Examples are the secure boot built into a microcontroller's boot ROM, which on one common Wi-Fi microcontroller uses RSA-PSS with 3072-bit keys, and MCUboot [MCUboot].

15.2 Verification on the Device

1. A device MUST verify the signature of a new image, and that the image is for its model and hardware revision, before it runs the image for the first time. It MUST refuse an image that fails, keep running its current firmware, and record `firmware-refused`.
2. This holds however the image arrives: over the module bus [OAP-MB Section 13], over the network, or over a wired connection by the owner.
3. Devices SHOULD use secure boot: verification of the firmware at every start by code that cannot be changed, against keys fixed in hardware, such as eFuses.
4. Verification on the device does not rely on any check by the controller. The integrity check of Module Bus Section 13.4 remains in addition.

15.3 Anti-Rollback

1. Each image has a security version, an integer that its manufacturer raises only when the image fixes a vulnerability that an attacker could exploit by installing an older image.
2. A device MUST refuse an image whose security version is lower than the highest security version it has confirmed.
3. A device MUST raise its stored security version only after the new image has been confirmed (Section 15.4).
4. Manufacturers SHOULD raise security versions rarely, because some hardware allows only a few raises (16 on one common Wi-Fi microcontroller), and because owners can freely return to any earlier image with the same security version.

15.4 Keeping the Previous Firmware

1. A device **MUST** keep its previous firmware until new firmware has started and confirmed that it works, as Module Bus Section 13.5 already requires of modules.
2. New firmware **MUST** confirm itself only after it has:
 - started and passed its own checks;
 - for a device with a controller, received a valid message from its controller again [OAP-FS Section 7.1]; and
 - for a controller, loaded its configuration, started its control functions, and regained contact with the devices it controlled before the update, or 5 minutes have passed with its control functions running (provisional).
3. If new firmware has not confirmed itself within 10 minutes of starting (provisional), or the device restarts before it has, the device **MUST** return to its previous firmware, and **MUST** record `firmware-reverted`.
4. A user with the `install-firmware` permission **MAY** return a device to an earlier image whose security version allows it.

15.5 Installing Updates

1. **Never forced.** A device **MUST NOT** install an image unless a user with the `install-firmware` permission approved it, or such a user turned on automatic installation for that device and channel [ETSI 303 645 provisions 5.3-4B and 5.3-6A]. Automatic installation **MUST** be off until it is turned on, and users with that permission **MUST** be able to turn it off, and to postpone an update, at any time. Automatic installation also needs an owner's consent to update checks (Section 16.2).
2. **Never in the middle of something.** An update **MUST NOT** start while an endpoint is dosing, filling, or taking part in a water change [OAP-FS Section 11.2]. Automatic installation **SHOULD** happen only within a time window the owner chooses.
3. **Informed.** Before an update, the controller **MUST** show the version, the channel, the release notes, whether it fixes security problems, how long the device will not be under control (the manifest's `interruption`), and what the device's outputs do meanwhile [OAP-FS Section 11.2].
4. **Offline.** A device **MUST** accept updates delivered on the local network, with no internet connection. Manufacturers **MUST** publish their images and manifests for download, so that owners can update offline and keep copies.
5. **Nothing taken away.** A manufacturer **MUST NOT** make local functions depend on installing an update, and **MUST NOT** disable a device that has not been updated.
6. **Recorded.** Every installation, refusal, and return to earlier firmware is recorded (Table 8).

15.6 Update Manifests and Feeds

A manifest tells a controller and its owner what a release is, and lets the controller check it before sending it to a device.

1. An update manifest is a JSON object [RFC 8259] with the members of Table 10, valid against `schema/firmware-update.schema.json` [JSON Schema].
2. A signed manifest is a JWS Compact Serialization [RFC 7515] whose payload is the manifest and whose protected header has exactly these members:
 - `alg`: `ES256` [RFC 7518] or `Ed25519` [RFC 9864], whose keys are JWKs of type `OKP` [RFC 8037];
 - `kid`: the JWK thumbprint [RFC 7638] of the signing key; and
 - `typ`: `oap-update-manifest`.
3. Controllers and apps **MUST** support `ES256`, and **MAY** support `Ed25519`. They **MUST** refuse a JWS with any other header member, any other algorithm, or a key they do not have.
4. A manufacturer publishes an update feed for each model at an HTTPS URL: a JSON object with `oapUpdateFeed("0.1")`, `manufacturer`, `model`, `updated` (a timestamp), and `releases`, an array of signed manifests.
5. Within a feed, versions are unique, and a release **MUST NOT** have a lower security version than any release with a lower version.
6. Before it sends an image to a device, a controller **MUST**:

- verify the manifest's signature with one of the update keys the device reports (Section 15.9);
 - check that the manifest's `manufacturer` and `model` are the device's, that its `hardwareRevisions`, if present, include the device's, and that the device's version is at least the manifest's `minimumVersion`;
 - refuse an image whose security version is lower than the device's; and
 - check the image's size and SHA-256 hash against the manifest.
7. Manifests are signed one by one, so that anyone, including the owner, can mirror or archive them.

Table 10 — Members of an update manifest

Member	Type	Presence	Description
<code>oapUpdateManifest</code>	string	REQUIRED	The revision of this format: "0.1".
<code>manufacturer</code>	string	REQUIRED	As the device reports it [OAP-DC Section 8.1].
<code>model</code>	string	REQUIRED	As the device reports it.
<code>hardwareRevisions</code>	list of strings	OPTIONAL	The hardware revisions the image is for. All of the model's, if absent.
<code>version</code>	string	REQUIRED	The firmware version, in Semantic Versioning 2.0.0 [SemVer].
<code>securityVersion</code>	integer	REQUIRED	The image's security version (Section 15.3).
<code>channel</code>	<code>stable</code> or <code>beta</code>	REQUIRED	Section 15.7. A <code>stable</code> release has no pre-release part in its version.
<code>released</code>	timestamp	REQUIRED	When the release was published.
<code>image</code>	object	REQUIRED	<code>size</code> in bytes and <code>sha256</code> , the image's SHA-256 hash in lowercase hexadecimal, both REQUIRED; <code>url</code> , an HTTPS URL to download it, and <code>format</code> , a name for its format such as <code>mcuboot</code> , both OPTIONAL.
<code>minimumVersion</code>	string	OPTIONAL	The lowest version this release can be installed over.
<code>interruption</code>	number (s)	OPTIONAL	The longest time the device is not under its controller's control while the release is installed, including its restart.
<code>securityFixes</code>	list	OPTIONAL	The vulnerabilities the release fixes, each with an <code>id</code> , such as a CVE identifier, and a <code>summary</code> .
<code>notes</code>	string	OPTIONAL	Release notes, as plain text.
<code>notesUrl</code>	string	OPTIONAL	An HTTPS URL of fuller release notes.
<code>supportEnds</code>	date	OPTIONAL	The date until which the manufacturer provides security updates for the model (Section 18.2).
<code>extensions</code>	object	OPTIONAL	Extensions, as in the Data Model [OAP-DM Section 7.6].

15.7 Channels

1. Releases are on the `stable` channel, for general use, or the `beta` channel, for owners who want to test.
2. A device's `updateChannel` MUST be `stable` until a user with the `install-firmware` permission chooses `beta` for it.
3. Beta releases are signed like stable ones.
4. When a device moves from `beta` to `stable`, the controller offers the latest stable release, which MAY be lower than the beta version if the security version allows it.

15.8 Owner-Installed Firmware

Owners who build or obtain other firmware for their equipment, for example after its manufacturer stops supporting it, need a way to run it.

1. A device SHOULD provide a documented way for its owner to run firmware that its manufacturer did not sign (Appendix D, issue 6). The reference designs MUST provide one.
2. Allowing it MUST need both an owner's permission (operation `allow-owner-firmware`) and physical presence. It MAY erase the device's credentials and owner data.
3. A device running such firmware MUST report `firmwareSigner owner` (Section 15.9), and controllers MUST show this.
4. Owner-installed firmware MUST still meet the rest of this Specification to be conforming, and a manufacturer is not responsible for it.

15.9 Firmware Information

A device reports what a controller needs to keep it up to date, as the device attributes of Table 11 [OAP-DC Section 8.1].

Table 11 — Firmware information

Device attribute	Type	Presence	Description
<code>updateChannel</code>	stable or beta	REQUIRED	The device's update channel (Section 15.7).
<code>securityVersion</code>	integer	REQUIRED	The highest security version the device has confirmed (Section 15.3).
<code>firmwareSigner</code>	manufacturer or owner	REQUIRED	Who signed the running firmware (Section 15.8).
<code>updateFeed</code>	string	OPTIONAL	The HTTPS URL of the model's update feed.
<code>updateKeys</code>	JWK Set	OPTIONAL	The public keys that sign the model's manifests [RFC 7517], each with its thumbprint as <code>kid</code> . REQUIRED if <code>updateFeed</code> is present.
<code>supportEnds</code>	date	OPTIONAL	The end of the model's security support (Section 18.2).
<code>automaticInstall</code>	boolean	OPTIONAL	Whether automatic installation is on (Section 15.5.1).
<code>lastUpdate</code>	object	OPTIONAL	The outcome of the device's latest update, <code>installed</code> , <code>reverted</code> , or <code>refused</code> , with the version, the time, and a reason (Section 15.5.6).

1. These are device attributes of Device Classes revision 0.2 and later, and a device that updates its firmware as this Specification defines MUST report those of them that are REQUIRED. A device of Device Classes revision 0.1 carries them instead in its device description, in `device.extensions` under the name `org.openaquariumproject.firmware`, as `channel` for `updateChannel`, `signer` for `firmwareSigner`, and the other members by the same names, without `lastUpdate`, valid against the `firmwareInformation` definition of `schema/firmware-update.schema.json`. A controller reads that extension only from a device of revision 0.1 [OAP-DC Section 7.11].
2. A controller MUST show its owners, for every device including each module: the manufacturer, model, hardware revision, firmware version, channel, signer, whether an update is available, and the end of security support.
3. `updateChannel` and `automaticInstall` are config attributes. A device MUST accept a change of either only from a party whose role permits `install-firmware` (Table 6), such as a write over the Device Protocol [OAP-DP Section 11.2], and a controller MUST make such a change only for a user whose role permits it.

15.10 Manufacturers' Keys

1. Manufacturers MUST keep the keys that sign images and manifests offline or in hardware security modules, apart from the servers that publish updates, so that a compromised server cannot sign anything.
2. Manufacturers SHOULD be able to replace a signing key: devices SHOULD accept more than one image key and support revoking one, as the secure boot of one common Wi-Fi microcontroller does with up to three, and new firmware MAY change a device's `updateKeys`.
3. A manufacturer whose signing key is compromised MUST say so publicly, and SHOULD release firmware that revokes it.

16 Privacy

16.1 Principle

The data a system collects belongs to its owner. Nothing leaves the owner's network without the owner's consent, except what Table 12 allows. This applies to devices, controllers, and apps alike.

16.2 What May Leave the Owner's Network

Table 12 — What may leave the owner's network

What	To whom	When
DHCP, DNS, ARP, and similar network traffic	The local network and the owner's DNS resolver	Always.
Time requests	Time servers; the owner can change which	Always, when the unit needs the time.
A request for the model's update feed	The manufacturer	Only after an owner has agreed to update checks. The request MUST NOT contain the device's serial number, key ID, or any other identifier of the unit, and MUST NOT carry cookies.
Image downloads	The manufacturer	When a user approves an update, or automatic installation is on.
Remote access traffic	The relay or VPN the owner chose	While remote access is on (Section 13).
Notifications	The notification services the owner chose	As the Alarms and Notifications specification defines.
What a service may see	A cloud service the owner granted access	Within its role (Section 13.4).
Telemetry and diagnostics	The manufacturer or anyone else	Only with the owner's consent (Section 16.3).

1. A unit or app MUST NOT send anything else off the owner's network.
2. A controller MUST ask an owner during setup whether to check for updates, and MAY recommend that they agree. It MUST NOT check before the owner has answered, and MUST NOT pre-select an answer.

16.3 Consent

1. Consent MUST be asked for separately for each purpose, such as update checks, crash reports, or usage statistics.
2. Before asking, the controller or app MUST say what data would be sent, to whom, how often, and how long it would be kept.
3. Every consent MUST be off until an owner gives it (operation `manage-consent`), and an owner MUST be able to withdraw it at any time, with effect at once.
4. Declining or withdrawing consent MUST NOT take away any function except the one that needs the data.
5. Consent MUST be recorded in the audit log. Firmware that adds a new purpose MUST ask again, and MUST NOT carry over consent given for another purpose.
6. A manufacturer that has received data MUST delete it when the owner asks, or when the owner withdraws consent, unless the law requires it to be kept.

16.4 What Telemetry May Contain

1. Telemetry MUST contain only what the owner agreed to.
2. It MUST NOT contain credentials, keys, Wi-Fi network names or passwords, names or notes the owner wrote, or the system's location more precise than its country, unless the owner explicitly agreed to each.
3. It SHOULD use an identifier that changes at least every 90 days, and not the unit's serial number or key ID, unless the purpose needs them, such as a warranty claim.

16.5 Network Destinations

1. A manufacturer MUST publish every internet host its product may contact, why, what it sends, and which setting controls it.
2. It SHOULD also publish them as a Manufacturer Usage Description [RFC 8520].
3. A unit MUST NOT contact any internet host that is neither on the published list nor one the owner configured, such as a relay, a VPN, a time server, or a notification service.

16.6 Export and Erasure

1. Users with the `export-data` permission MUST be able to export, on the local network and at no charge:

- history, as Data Model documents [OAP-DM];
 - the configuration and its state, as an Automation document [OAP-AU Section 19];
 - the audit log (Section 12.3); and
 - for owners, the users and pairing lists, without any secret.
2. An owner **MUST** be able to erase history and other personal data without a factory reset (operation `erase-data`), and with a factory reset erase everything (Section 9.8) [ETSI 303 645 provision 5.11-1]. Erasing data does not erase the audit log, except by factory reset (Section 12.3), but records the erasure in it.
 3. Before configurations are shared, controllers remove what identifies the owner, as Automation Section 21.6 describes.

17 The Module Bus

17.1 Physical Access Model

The module bus has no message authentication in this revision [OAP-MB Section 15]. It relies on physical access control: the bus runs inside the equipment cabinet, and anyone who can reach it can also unplug the equipment [OAP-FS Section 15]. `oap.controller.id` and `oap.controller.unique_id` prevent mistakes, such as two controllers on one bus, but not deliberate attacks.

17.2 What the Model Does Not Cover

The physical access model also trusts every node on the bus. A module running faulty or malicious firmware, such as an unofficial module or one with a vulnerability, could:

1. pose as the controller, which is node-ID 1, and command or reconfigure other modules;
2. send false readings in another module's name, for example a low temperature that makes the controller run a heater; or
3. offer other modules a firmware image of its own.

Signed firmware (Section 15.2) stops the third. The first two need message authentication (Section 17.4).

17.3 Requirements in This Revision

1. Modules **MUST** verify the signature of firmware images before running them (Section 15.2).
2. A controller **MUST** record each node that joins the bus for the first time (`node-joined`), and each node that uses its node-ID or acts as a controller (`node-conflict`), and **SHOULD** raise a fault for the second.
3. Reference designs **SHOULD** keep the module bus inside the cabinet or enclosure, and **SHOULD NOT** provide bus connectors where people outside the household could reach them.
4. Diagnostic tools on the bus, at node-IDs 126 and 127 [OAP-MB Section 9.2], count as physical presence.

17.4 Adding Message Authentication on CAN FD

This section is informative. It sets out how a later revision could authenticate the module bus on CAN FD, which has room for it. Classic CAN does not, so authenticated buses would need CAN FD.

What to authenticate. Commands, register writes, and ExecuteCommand requests from the controller; the controller's supervision; and endpoint state, events, and responses from modules.

Keys. Each module shares its own key with the controller: two 128-bit keys, one for each direction. A shared bus-wide key would let any module pose as the controller, which is the threat being addressed.

Tag. Each authenticated transfer would carry its message inside a wrapper data type, with the fields of Table 13. Appending the tag after the message instead would not work: the Open Aquarium data types are delimited, so a receiver that knows a newer minor version of a type would read the tag as new fields.

Table 13 — Proposed fields of an authenticated transfer

Field	Size	Contents
Message	As now	The serialized message, unchanged.

Field	Size	Contents
Freshness	4 bytes	The lowest 32 bits of a 64-bit counter kept for each key, which increases with every transfer sent with the key.
Tag	8 bytes	AES-128-CMAC [RFC 4493], truncated to 64 bits, over the port-ID, the source and destination node-IDs, the transfer kind, the full 64-bit counter, and the message.

The two fields add 12 bytes, and a little more for the wrapper's length field. A single 64-byte CAN FD frame carries up to 63 bytes of a transfer, so a message of up to about 50 bytes, such as most commands and the supervision messages below, still fits in one frame; longer ones take more frames, as they do now. The 5-bit Cyphal transfer-ID [Cyphal] is too short to prevent replay, which is why the freshness counter is needed.

Replay and restarts. Transfers sent at different priorities can arrive out of order, so a receiver keeps a window of recently accepted counter values, for example the last 64, and accepts a transfer whose counter is new and not older than the window. To avoid wearing out flash, a node stores its counter in steps, for example every 65 536 transfers, and after a restart continues from the next step.

Supervision. The controller's Heartbeat is sent to every module at once, so one tag cannot serve all of them without a shared key. Instead, the controller would send each module its own small authenticated supervision message once a second, which on a bus of 32 modules is about 32 frames a second. A module with authentication turned on counts only authenticated messages as contact [OAP-FS Section 7.1].

Key distribution. When a module first joins the bus, the controller and the module run the pairing of Section 9.4 over the bus, with the module's setup code, which the owner scans when installing it. Both derive the two keys from the SPAKE2+ shared secret with HKDF [RFC 5869]. A module would keep its keys until it is moved to another controller [OAP-MB Section 9.3] or factory reset. Where scanning codes is not wanted, the owner could instead confirm new modules on the controller after each identifies itself, which trusts whatever is on the bus at that moment.

Migration. Modules would report support in a register, and the controller would turn authentication on per module, which switches the module's ports to the wrapper type. Buses with and without authentication would work together, and a module with authentication on would refuse unauthenticated commands.

Cost. AES-CMAC is fast in hardware, which some small microcontrollers suited to modules have, and fast enough in software on the rest. Pairing needs one SPAKE2+ exchange per module, which published timings of similar operations suggest takes one to two seconds on a 64 MHz microcontroller without a public-key accelerator.

A later revision could require it only after a prototype shows how freshness survives restarts, what supervising each module costs, whether time synchronization needs authentication too, and how a module learns that authentication is required.

18 Manufacturer Obligations

18.1 Vulnerability Handling

1. A manufacturer **MUST** publish a vulnerability disclosure policy, with a contact for reporting security problems and how long it will take to acknowledge them and report progress [ETSI 303 645 provision 5.2-1] [PSTI].
2. It **MUST** fix vulnerabilities in a timely way, and publish what each security release fixes (`securityFixes`).
3. It **SHOULD** publish a software bill of materials for each release.

18.2 Support Period

1. A manufacturer **MUST** publish, when a product is sold, the date until which it will provide security updates for it [ETSI 303 645 provision 5.3-13] [PSTI], and report it as `supportEnds` (Section 15.9). It **MAY** extend the date, and **MUST NOT** shorten it.
2. When support ends, every local function **MUST** keep working.

3. A manufacturer SHOULD then let owners install firmware of their own (Section 15.8), and SHOULD keep its last images and manifests available.

18.3 Documentation

A manufacturer MUST document, for owners: how to claim, pair, share, revoke, and factory reset its product; what roles it supports; how remote and emergency access work on it; what data it may send and where (Section 16.5); and how to update it offline.

19 Conformance

19.1 Devices

A conforming device that connects to a network meets Sections 8 to 10, 11.3, 11.6, 12 as it applies to devices, 15, and 16. A conforming module meets Sections 15 and 17.3.

19.2 Controllers

A conforming controller meets Sections 8 to 17.

19.3 Apps

A conforming app meets Sections 8.1 to 8.4, 9.4, 10, 13.3, and 16.

19.4 Manufacturers

A manufacturer claiming conformance for a product meets Sections 15.10, 16.3.6, 16.5, and 18.

19.5 Machine-Readable Definitions and Tools

`permissions.json` is normative, and contains the same definitions as Tables 5, 6, and 8, which are generated from it by `tools/render_spec_tables.py`. The JSON Schemas [JSON Schema] `schema/firmware-update.schema.json` and `schema/audit-log.schema.json` are normative for the structure of the documents they describe. Where a schema and this document differ, this document governs.

The validator, `tools/validate_security_and_access.py`, checks update manifests, feeds and their signatures, key sets, and audit logs against the schemas and the rules a schema cannot express. `tools/sign_update_manifest.py` makes update keys and signs manifests. Both are informative.

The `examples` directory contains an update manifest, a signed update feed with three releases, the key set that verifies it, and an audit log.

19.6 Test Procedures

Tests use a controller and devices whose clocks and network the tester can set, and tools that capture network and radio traffic. Some tests inspect a product, its documentation, or what its manufacturer publishes, rather than run it. Each test names the requirements it checks. Every MUST and MUST NOT of Sections 8 to 18 is checked by at least one test. A pass criterion that checks what a SHOULD asks applies to a product that does it.

Table 14 — Test procedures

Test	Requirements	Method	Pass criteria
Identity keys	8.1, 8.2.3, 8.4.1, 8.4.3	Read the identity certificates of several units of one model, and of two installations of an app; try every interface of a unit, and its firmware's debug output, for a private key or stored secret; then present a unit with a certificate for a paired key that has expired dates and another name, and with a certificate for an unpaired key	Each key is an ECDSA P-256 key, and every key ID differs. No interface or output reveals a private key, credential, Wi-Fi password, or pairing verifier. The paired key is trusted whatever its certificate says, and the unpaired one is not. The app uses the platform's protected key store.

Test	Requirements	Method	Pass criteria
		that has the paired key's name. Inspect where the app keeps its key.	
No default credentials	9.1.1 to 9.1.3	Before claiming, try every network and wireless interface of the unit, including web and Bluetooth, with empty and common credentials, and with the manufacturer's service tools.	Nothing but pairing and discovery responds.
Unique setup codes	9.1.1, 9.2.1, 9.2.4	Compare the setup codes and verifiers of several units of one model, and read a unit's storage.	All differ, none can be derived from a serial number or network address, none is a repeated digit, 12345678, or 87654321, and no unit stores its code in a form that can be read back.
Setup code labels	9.2.2	Inspect a unit, its packaging, and, for a unit too small for a label, its documentation.	The setup code is printed as digits and as a QR code on the unit and its packaging, or on the packaging and in the documentation.
Pairing window	9.2.4, 9.3.1, 9.3.3, 9.3.4	Try pairing 15 minutes after start-up; then make 20 failed attempts in a new window; then start two pairing exchanges at once. Create an invitation valid for 2 days, pair with it after 1 day, and try it again. Use a claimed unit's setup code without physical presence.	The first two are refused until a button press opens a new window. The second of two exchanges at once is refused. The invitation works after a day, and only once. The setup code does nothing on a claimed unit without physical presence.
Pairing binds keys	9.4, 9.6.2	Run a pairing through a proxy that terminates TLS and opens its own connection to the unit; then pair as Provers with verifiers of 1000 and of 100 000 iterations.	The pairing through the proxy fails. Both other pairings succeed.
Bluetooth setup	9.4.6, 9.6.2, 9.6.3	Claim a unit and give it Wi-Fi settings over Bluetooth, capturing the radio traffic; then look for its Bluetooth advertising, and connect to it over Bluetooth.	The pairing exchange of Section 9.4 runs with both identity keys, with no Bluetooth pairing, and protects everything after it. Once the unit has joined its network, it does not advertise and accepts no Bluetooth connection, until a button press allows new network settings.
Claiming	8.5.2, 9.5.3 to 9.5.6, 11.7.2	As a user without <code>add-device</code> , ask the controller to claim a device; then as a user with it, claim a device built by its owner, which cannot prove who made it. Claim another device from an owner's app, for a system with a controller. Add an owner credential, then revoke it.	The first is refused. The device built by its owner is claimed. Each device lists its controller, marked as its controller, and an <code>owner</code> entry for each owner, with two entries free. The new credential is added to each device's pairing list, and removed when it is revoked. Each claiming is recorded as <code>claimed</code> on the controller, and on each device that keeps an audit log.
Factory reset	9.8.1 to 9.8.5	Factory reset a claimed device by physical presence, after setting it to refuse resets if it offers any such setting, then inspect it through every interface; watch its outputs during and after the reset.	The reset happens. No owner data or credential remains; it accepts only pairing; its key ID has changed, if it replaces its identity key (Section 8.5); its fail-safe settings are the manufacturer's; its outputs behave as after a power-up.
Removing a device	9.9.2	Remove a device from the system.	The controller offers to factory reset it.
Physical presence	9.10.1, 9.10.2, 14.2.3	Remove and restore the power of a claimed unit, and send commands through its wired service connection, without pressing a button; look for the wired service connection from the network.	Neither opens a pairing window, recovery, emergency access, or owner-installed firmware. The wired service connection cannot be reached from any network.
TLS profile	10.1.1, 10.1.2, 10.2.1, 10.2.2	Connect to a unit offering only TLS 1.2; then with only <code>TLS_AES_128_GCM_SHA256</code> , <code>secp256r1</code> , and <code>ecdsa_secp256r1_sha256</code> ; then without a client certificate. Connect from the unit to a party that requests a certificate.	TLS 1.2 is refused. The mandatory suite, group, and scheme work. A connection without a client certificate is refused, except by a web interface. The unit sends its identity certificate.

Test	Requirements	Method	Pass criteria
Plain traffic	10.1.1, 13.1.3	Capture all traffic of a claimed system in normal use, including starting and ending modes and overrides, locally and through a relay.	Nothing but discovery, an HTTP redirect, and the traffic of Table 12 is unencrypted.
Unpaired key	10.3.1, 10.3.2	Connect with a valid TLS client that presents an unpaired identity key, to a controller, a device, and an app.	Every request is refused, except pairing and password sign-in.
Commands from other keys	10.3.3, 10.3.4	From an owner's app that is not the device's controller, send a device supervision messages, a command other than <code>identify</code> , and <code>control.take</code> ; then write a setting that its role permits.	The supervision and the command are refused and do not count as contact; <code>control.take</code> is refused; the write is accepted.
Early data	10.2.3	Send a command as TLS early data.	It is not carried out.
Web interface	10.1.1, 10.4.1 to 10.4.5	Connect to the web interface over HTTP; send a request with a <code>Host</code> header naming another domain, one from another origin, and a change without the protection against cross-site request forgery; inspect the session cookie, and leave a session unused for 30 minutes.	HTTP only redirects to HTTPS, served with the unit's identity key. The three requests are refused. The cookie is <code>Secure</code> , <code>HttpOnly</code> , and <code>SameSite=Strict</code> , and the session has ended.
Web interface through an app	10.4.6	Open a controller's web interface in a browser through an app that relays it. From another program on the same computer, connect to the app's port without the secret, and with a <code>Host</code> header naming another domain; from a web page of another origin, send a change; from another computer, connect to the port. Then close the interface, and, after opening it again, revoke the app's credential.	The browser is served the interface as the app's user, with no password, and never connects to the controller. The program without the secret, the other <code>Host</code> , and the change from the other origin are refused; the port cannot be reached from the other computer. The app stops relaying when the interface is closed, and when its credential is revoked.
Password limits	10.5.1 to 10.5.4	Try to set a password of 14 characters, one from a list of compromised passwords, and one of 64 characters; inspect what the unit stores; capture a sign-in; then make repeated wrong sign-ins for one user.	The first two are refused, and the third accepted. Only a salted hash of the required strength is stored. The password travels only inside TLS. No more than 100 guesses in 24 hours are evaluated, and failures are recorded.
Revocation	10.2.4, 10.6.1, 10.6.2, 11.4.2	Revoke a credential while it has an open connection; let a grant expire; remove a key from a device's pairing list; then try to resume each session.	Each connection closes within 5 s, and new connections and resumed sessions are refused. The expiry is recorded as <code>grant-expired</code> .
Invalid clock	10.7.1, 10.7.2, 11.8.4	Restart the controller with its clock not valid.	Owners and devices authenticate; users with time-limited grants can use only <code>view-state</code> , <code>stop-all</code> , and overrides that stop an output; <code>clock-invalid</code> is recorded.
Network exposure	10.8.1 to 10.8.4, 10.9.3, 13.2	Scan every port of a unit, before and after claiming; capture its traffic to the router; read its discovery records; inspect its production firmware and its Wi-Fi settings.	Only the ports and services that this Specification and the Device Protocol need answer. The unit asks the router to open no port. Discovery records reveal no more than discovery needs. The firmware has no debugging or update service that bypasses this Specification. The unit supports WPA2-Personal, and does not require an open network.
Flood	10.9.1, 10.9.2	Flood the controller with connection attempts and traffic for 10 minutes.	Control functions keep running, devices stay supervised, and a paired app can still connect.
Safety function paths	10.9.4	Bind a heater to an outlet reached over Wi-Fi whose <code>failSafeAction</code> is <code>continue</code> , with no thermostat of its own; then change its fail-safe configuration to stop on loss of contact.	The controller accepts the configuration, and shows every owner a standing warning that names the heater, until the change, which ends it.

Test	Requirements	Method	Pass criteria
Users	11.1.1, 11.1.3, 11.1.4, 11.6.2, 11.6.5	Create 16 users with 32 credentials in all; try to give a service the <code>owner</code> role, and to use emergency access as a service; try to remove or demote the last owner; fill a device's pairing list with 16 entries, then try to remove its last <code>owner</code> entry.	Every user and credential works. The service is refused both. The controller refuses to remove or demote the last owner. The device keeps 16 entries, and refuses to remove its last <code>owner</code> entry.
Roles	11.3.1, 11.3.2, 12.1.1	For each role, attempt every operation of Table 6, on the controller and on a device.	Exactly the operations the role permits succeed. Refusals of audited operations are recorded on the controller, and on a device that keeps an audit log.
Invitations	11.5.3, 11.5.5	As an owner, create invitations with no expiry given and with an expiry of 8 days; pair with one; try to use it again; leave another unused for 24 hours.	The first expires after 24 hours, and the second is refused or limited to 7 days. The pairing adds a credential, which every owner is shown, and which is recorded as <code>paired</code> ; the invitation does not work again.
Owner recovery	11.7.3	Revoke every owner credential, then press the controller's button and pair with its setup code; repeat with recovery turned off.	The first gives a new owner and records <code>owner-recovery</code> ; the second is refused.
Stop all	11.8.1 to 11.8.6, 14.2.1, 14.5.1	As a user with the <code>override</code> role, use <code>stop-all</code> : normally, while the clock is not valid with a time-limited grant, in an emergency session, while a module is updating its firmware, with an output under local control, and from the controller's own buttons with no credential. Include a network device, a module, a dosing head that runs its own program, a feeder, a module that does not respond to commands, a siren on a module and another on a network device, each driven by an alarm function, and a return pump that an interlock holds with a latched stop. Ask for an expiry of 25 hours. Let one <code>stop-all</code> expire while the controller is stopped.	Every output but the sirens stops at once each time, and stays stopped until <code>stop-all</code> expires or is ended, at the precedence of interlocks; the sirens follow their alarm function, and sound for the critical alarm <code>stop-all</code> raises. Each stop carries <code>revertAfter</code> for the time left, except the <code>stop</code> of the dosing head and the feeder, and the head is also held with <code>set</code> , <code>enabled false</code> , and <code>revertAfter</code> , whichever goes first. Modules that respond get no <code>COMMAND_EMERGENCY_STOP</code> ; the one that does not gets it 1 s after its stops; the module and the network device with a siren get neither <code>COMMAND_EMERGENCY_STOP</code> nor <code>device.stop</code> . The expiry of 25 hours is refused. With the controller stopped, each device's revert falls due at the expiry, no dose or feed resumes, and the latched return pump stays stopped.
Audit log	9.5.6, 12.1.1, 12.1.2, 12.3.1 to 12.3.4, 16.3.5	Perform each audited operation and event, including a change of a fail-safe setting and a consent; restart the controller with its clock not valid; fill the log past 2000 records; try to change or delete a record; then export it.	Each is recorded with the members of Table 7, the fail-safe change with its old and new values; <code>seq</code> keeps increasing across the restart; the oldest records are dropped first, and at least 2000, or 90 days, are kept; no record can be changed or deleted; the export validates, and holds no password, code, or private key.
Matter fabrics	12.1.5, 12.1.6, 13.4.4, 13.4.5	As a user with <code>full-control</code> , try to turn on a controller's Matter bridge, to open its commissioning window, and to remove a fabric. As an owner, turn the bridge on with the <code>override</code> role, open its commissioning window, and commission it into one ecosystem as an administrator; from that ecosystem, open a commissioning window, and commission a second ecosystem; set an override of an output from each, and give each fabric administrator privilege in Matter's access control; then remove the first from an Open Aquarium app.	The <code>full-control</code> user is refused each time. <code>fabric-added</code> is recorded for each ecosystem, and <code>fabric-removed</code> for the first, and every owner is notified of each. Each override succeeds, and nothing beyond the <code>override</code> role does. Every record of the bridge's actions has the bridge's user as <code>actor</code> , <code>via matter</code> , and <code>fabricIndex</code> , <code>label</code> , and <code>vendorId</code> in <code>details</code> .
Remote access off	13.1.2, 13.1.5	Capture the controller's traffic before remote access is turned on, and try to connect to it from the internet.	It makes no connection to any relay, and accepts no connection from the internet.

Test	Requirements	Method	Pass criteria
Remote access	13.1.1, 13.1.2, 13.1.4, 13.1.7, 13.1.8	Turn remote access on; connect remotely as an owner and as a user whose grant does not allow it; do everything the owner did remotely again on the local network; then cut the connection to the relay while the controller runs; turn remote access off.	The owner's session is recorded as <code>remote-session</code> , and the other user is refused. Everything works locally. Control goes on when the relay is lost. Remote access stops within 5 s.
Remote access end to end	13.1.3, 13.3.1 to 13.3.3	Connect through a relay that records what it forwards; make the relay present another key ID for the controller.	The relay holds only TLS records between app and controller; the controller authenticates to the relay with its identity key; the app refuses the wrong key.
Services	13.4.1 to 13.4.3	Give a service the <code>owner</code> role; connect a cloud service and a home automation service, then revoke the cloud service from an app on the local network while it is connected.	The <code>owner</code> role is refused. Local sessions and integrations work while the cloud service is connected; its access ends, and it cannot prevent it.
Emergency access	14.3.1, 14.3.2, 14.4.1, 14.4.2, 14.5.1, 14.5.3	Disconnect the controller from the home network for 2 minutes; connect over the direct link as an owner, then with an emergency code, which no pairing window allows; try to change the configuration, and, as the emergency code's user, to add a credential. Then try the emergency code over the home network, and over the direct link before it is available.	The link appears after 2 minutes; both sessions are emergency sessions, and the app is told so; overrides succeed; the configuration change and the new credential are refused; each session's start and end, and each operation, are recorded with <code>via emergency</code> . The code is refused over the home network and before the link is available.
Emergency access limits	14.3.1 to 14.3.3, 14.6.1, 14.6.2	Before emergency access is turned on, look for the direct link while the home network is down. Turn it on, and look for the link while the home network works, without pressing a button on the controller. Disconnect the home network for 2 minutes, join the link, and try to reach any other network or device; then leave the link unused for 30 minutes. On a controller that offers emergency access on its device network, join that network with an owner's app while the home network works, try to open an emergency session and any other session, and try to reach a device on it.	The link is not available before it is turned on, nor while the home network works. When it is available, it uses WPA2-Personal or WPA3-Personal with a passphrase unique to the unit, nothing but emergency access on the controller can be reached, and it closes 30 minutes after it was last used. On the device network, both sessions are refused, and no device can be reached.
Emergency code lockout	14.4.3	Make 10 failed attempts with emergency codes, then try the right code.	It is refused until a button press on the controller or an owner allows codes again.
Signing	15.1.2, 15.10.1, 15.10.3	Inspect the manufacturer's signing scheme, where it keeps its signing keys, and its policy for a compromised key.	Signatures give at least 128-bit security; the keys are offline or in hardware security modules, apart from the update servers; the policy says a compromise will be announced publicly.
Unsigned image	15.1.1, 15.2.1, 17.3.1	Offer a device, and a module, an image that is unsigned, signed with another key, for another model, or changed after signing.	Each is refused, the device keeps running, and <code>firmware-refused</code> is recorded.
Anti-rollback	15.3.2, 15.3.3	Offer a device an image with a higher security version that never confirms itself, then an older image; then confirm an image with a higher security version, and offer the older image again.	The older image is accepted while the higher version is not confirmed, and refused after it is.
Failed update	15.4.1 to 15.4.3	Install an image that never confirms itself; separately, cut power partway through an update; separately, install an image on a device whose controller is stopped.	The device returns to its previous firmware, and <code>firmware-reverted</code> is recorded; the last image does not confirm itself until the controller sends a valid message.

Test	Requirements	Method	Pass criteria
Never forced	15.5.1, 15.5.3, 15.7.2	Publish an update in the feed with automatic installation off, and a beta release; then approve the stable update.	Nothing is installed until a user approves it; the device stays on <code>stable</code> and is not offered the beta; before installing, the controller shows the version, channel, release notes, security fixes, <code>interruption</code> , and what the outputs do meanwhile.
Update during dosing	15.5.2	Start an update while an endpoint is dosing.	The update does not start.
Offline update	15.5.4	With no internet connection, update a device from an image and manifest saved earlier.	The update installs.
Manifest checks	15.6.3, 15.6.5, 15.6.6	Give the controller manifests signed with <code>ES256</code> , with a bad signature, for another model, with a lower security version than the device's, and whose image does not match its hash; check the manufacturer's feed with the validator.	The <code>ES256</code> manifest is accepted, and the controller refuses to send the others to the device. The feed passes, with unique versions whose security versions never fall.
Owner-installed firmware	15.8	On a reference design, allow owner firmware without physical presence, and then with it; install firmware the manufacturer did not sign; then run the other tests of this table on the device.	It is refused without physical presence. The device runs the firmware, reports <code>firmwareSigner owner</code> , and the controller shows it. The device is conforming only if it passes the other tests.
Firmware information	11.3.2, 15.9	Read a device's firmware information, then change its <code>updateChannel</code> as a user without <code>install-firmware</code> , and as one with it.	The device reports <code>updateChannel</code> , <code>securityVersion</code> , and <code>firmwareSigner</code> . The controller shows, for every device and module, the manufacturer, model, hardware revision, firmware version, channel, signer, any available update, and the end of security support. Only the second change is accepted.
Update check privacy	16.2, 16.2.1	Capture an update check.	The request contains no serial number, key ID, or cookie.
Telemetry	16.2.1, 16.4.1, 16.4.2, 16.5.3	Capture the traffic of a unit with all consents withdrawn, over 24 hours; then give one consent and capture what is sent.	Nothing outside Table 12 leaves the network, and no host is contacted that is neither published nor configured. With consent, only what the owner agreed to is sent, with no credential, key, network name, owner's note, or precise location.
Consent off by default	16.2.2, 16.3.1 to 16.3.4	Set up a new controller, decline or skip every question about data, and capture its traffic for 24 hours; then give a consent and withdraw it.	The controller asks about update checks, with no answer chosen, and asks for each purpose separately after saying what would be sent, to whom, how often, and for how long. Nothing leaves the network except the first three rows of Table 12. Withdrawing consent stops what it allowed at once, and takes away no other function.
Export and erasure	16.6.1, 16.6.2	As a user with <code>export-data</code> , export everything on the local network; then, as an owner, erase personal data.	The export holds history, configuration, and the audit log, with users and pairing lists, without secrets, for an owner. History and personal data are erased, the audit log is kept, and the erasure is recorded.
Module bus nodes	17.3.2	Connect a new module; then connect a node that uses the controller's node-ID.	<code>node-joined</code> and <code>node-conflict</code> are recorded.
Manufacturer obligations	15.5.5, 16.3.6, 16.5.1, 18.1.1, 18.1.2, 18.2.1, 18.2.2, 18.3	Inspect what the manufacturer publishes and documents, its process for deleting data and for fixing vulnerabilities, and a product whose support has ended or whose update was not installed.	A vulnerability disclosure policy, every internet host the product may contact, the support period, and the documentation for owners are published; security releases list what they fix; data is deleted when an owner asks; every local function works without the update, and after support ends.

20 Security Considerations

This whole Specification is about security. This section lists the risks that remain.

1. **Labels.** A setup code on a label can be read by anyone who sees the device, or a photo of it. It is useful only within a pairing window, which needs physical presence, and only until the unit is claimed, or on a controller for owner recovery. Units with displays avoid the label altogether (Section 9.2.3).
2. **Stolen verifiers.** Anyone who reads a unit's stored pairing verifier can find an eight-digit code by trying every value offline, which is quick. Encrypting secrets at rest (Section 8.4) and factory resetting second-hand equipment limit this.
3. **Online guessing.** With 20 attempts per window, an attacker who can open windows has about one chance in five million per window of guessing a code.
4. **Browsers.** A first visit to a web interface can be intercepted by someone on the local network, who can then capture a password (Section 10.4). Apps that check keys, and relay the interface to a browser on the same phone or computer, avoid this (Section 10.4.6).
5. **Relays.** A relay learns when, how often, and from where the owner connects, and how much data flows. Owners who mind can run their own relay or a VPN.
6. **Recovery.** Someone with physical access and the controller's setup code can become an owner through recovery and see the system's history. Owners who are worried about this can turn recovery off (Section 11.7).
7. **Owner-installed firmware** has none of the manufacturer's protections, and an attacker with physical presence and an owner's permission could install malicious firmware. Controllers show such devices (Section 15.8.3).
8. **Denial of service** cannot be prevented. Equipment that loses contact takes its fail-safe behavior, which is safe by design, and safety functions keep a path that uses no radio link (Section 10.9.4).
9. **The module bus** is unauthenticated, so a malicious module can command other modules (Section 17.2).
10. **A compromised controller** controls its whole system. Keeping controller firmware up to date, offering few network services, and using a separate network for devices [P0001] all reduce the risk.
11. **Time.** An attacker who can change an unauthenticated clock can shift schedules and make grants last longer. Battery-backed clocks and authenticated time limit this (Section 10.7).
12. **Long service lives.** Aquarium equipment can stay in service for ten years or more, longer than some algorithms may stay safe. Manufacturers should be able to change firmware signing algorithms, for example to hash-based signatures [RFC 8554].

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	
An app and a controller from different implementations have paired, shared access, and updated a device from a third.	Not started	
The implementations have passed the test procedures in Section 19.6.	Not started	
An independent security review of this Specification and one implementation has been completed.	Not started	
Every normative reference is publicly available.	Done	All are free.
The permissions file, schemas, and examples are published, and the examples pass the validator.	Done	

Item	Status	Details
The Open Issues appendix is empty, or each open issue is explicitly deferred.	In progress	Issue 14 waits on the bench tests of the breadboard controller; issues 10, 15 to 22, 26, and 27 are deferred beyond revision 1.0.
No unresolved Exclusion Notices for this Specification are recorded in <code>Notices.md</code> .	Done	
The Maintainer has recorded consensus to approve.	Not started	

Appendix B — Flows (Informative)

B.1 Claiming a Controller and Adding a Device

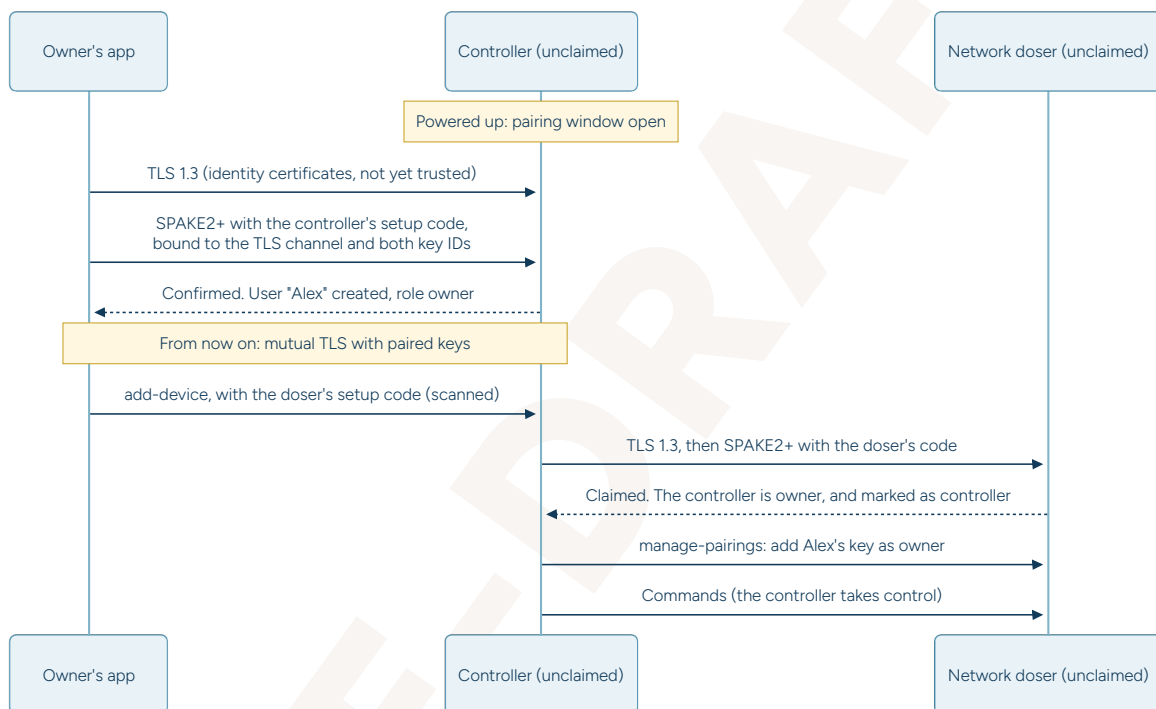


Figure 2 — Claiming a controller, and adding a network device through it

B.2 Sharing with a House-Sitter

- Alex, an owner, creates an invitation for “Sam” with the `override` role, an expiry on Sunday evening, and no remote access.
- The controller shows an eight-digit code and a QR code, valid for 24 hours, and Alex sends it to Sam.
- Sam’s app, on the home network, pairs with the controller using the invitation (Section 9.4). The controller records `paired` and shows Alex the new credential.
- During the week, Sam starts feed modes and sets overrides; each is recorded.
- On Sunday evening, the grant expires: Sam’s connections close, and the controller records `grant-expired`.

B.3 A Lost Phone

- Alex signs in on their laptop’s app, which is another credential of the same user, and revokes the phone’s credential (`manage-own-credentials`).
- The controller closes the phone’s connections, and removes the phone’s key from the pairing lists of the system’s devices (Section 11.7).

3. If Alex had no other credential, another owner could revoke it, or Alex could use owner recovery at the controller.

B.4 Replacing a Failed Controller

1. The old controller has failed. Its devices have lost contact, and their outputs have taken their fail-safe behavior.
2. Alex claims the new controller (B.1).
3. For each network device, Alex's app, which has the `owner` role in the device's pairing list (Section 9.5.4), adds the new controller with the `owner` role, marked as the device's controller, and removes the old one. Devices that were on the old controller's device network [OAP-DP Section 14.1] have lost their network too, so the app cannot reach them. Alex presses each one's button and gives it new network settings with its setup code (Section 9.6.4), where the device supports it, and otherwise factory resets the device and claims it again.
4. The new controller takes control of each device, as Fail-Safe Behavior Section 7.5 allows after the loss of contact.
5. Modules on the module bus see a controller with a different unique ID, clear their node-ID, port-IDs, and controller, keep their fail-safe configuration, and are configured by the new controller [OAP-MB Section 9.3].
6. Alex imports the old configuration, if it was exported, and binds it to the devices [OAP-AU Section 19].

Appendix C — Consumer IoT Security Provisions (Informative)

How this Specification relates to some provisions of ETSI EN 303 645 [ETSI 303 645], which several regulations build on. Meeting this Specification does not by itself show compliance with any regulation.

Table 15 — ETSI EN 303 645 provisions

Provision	Topic	In this Specification
5.1-1	Passwords unique per device or defined by the user	Section 9.1
5.2-1	Public vulnerability disclosure policy	Section 18.1
5.3-4B, 5.3-6A	Automatic updates after the user's consent; users can turn them off and postpone them	Section 15.5
5.3-9, 5.3-10	Updates checked for authenticity and integrity	Sections 15.2 and 15.6
5.3-13	Published support period	Section 18.2
5.6-1	Unused interfaces disabled	Section 10.8
5.11-1	Users can erase their data	Sections 9.8 and 16.6

Appendix D — Open Issues (Informative)

Issues 1 to 31 were open in revision 0.1. The Project Lead decided issues 5, 32, and 33 on 15 October 2026, accepting the recommendation in each. Issues 34 and 35 arose in revision 0.2, and the Project Lead decided both on 16 October 2026, accepting the recommendation in each. On 17 October 2026 the Project Lead froze the scope of revision 1.0, and decided issues 1 to 4, 6, 13, 23, and 28 to 31; revision 0.3 applies the decisions, and resolves issues 8, 9, 24, and 25, which were already answered. Issue 36 records the choices that revision 0.3 made where those decisions left them open, which the Project Lead accepted on 18 October 2026. Issues 10, 15 to 22, 26, and 27 are deferred beyond revision 1.0, and are tracked outside this Specification. Issue 14 stays open: its numbers wait on the bench tests. Each issue says how a revision resolves it, or why it stays open. Issue numbers do not change, because other specifications refer to them.

1. **TLS on the local network.** *Resolved in revision 0.3, as the Project Lead decided on 17 October 2026.* TLS 1.3 with mutual authentication stays mandatory for every connection, battery-powered devices on Thread included (Section 10.1): it is achievable on the reference controller's microcontroller, and one protocol

everywhere keeps implementations simple. Revision 0.2 let battery devices on Thread use DTLS 1.3 provisionally, and revision 0.3 removes that. A profile for battery devices on Thread, with DTLS 1.3 [RFC 9147], or OSCORE with EDHOC [RFC 8613] [RFC 9528], is deferred beyond revision 1.0, and plain connections for reading only, like the unprotected local interfaces of Table 1, are not allowed.

2. **Pairing method.** *Resolved in revision 0.3, as the Project Lead decided on 17 October 2026: SPAKE2+.* SPAKE2+ (Section 9.4) is an RFC, and Matter uses it in an earlier draft form. The alternatives were SRP-6a, which one microcontroller maker's provisioning software already implements but which is older and slower, and CPace [CPace], which is not yet an RFC.
3. **Default remote access.** *Resolved in revision 0.3, as the Project Lead decided on 17 October 2026.* The default is one open relay protocol for owner-run relays and relay services, which the Device Protocol defines (its Section 15.4), with VPNs as the alternative that needs nothing from the controller (Section 13.2). A relay works behind carrier-grade NAT, and needs no router settings.
4. **Web interfaces.** *Resolved in revision 0.3, as the Project Lead decided on 17 October 2026.* Browsers cannot check paired keys (Section 10.4). A browser reaches a unit's web interface through a paired app, which serves the interface on the loopback address and relays its requests over a connection that it authenticates with its identity key (Section 10.4.6), as the Device Protocol defines (its Section 15.5). An owner can still reach a web interface directly, and compare its certificate's fingerprint with the one a paired app shows (Section 10.4.1). A certificate service, which would need the internet and a service run for the project, is deferred beyond revision 1.0.
5. **Bluetooth after setup.** *Decided by the Project Lead on 15 October 2026; resolved in revision 0.2.* Revision 0.1 allowed emergency access over Bluetooth provisionally, which would have extended the decision that Bluetooth is for setup only. The Project Lead decided that Bluetooth stays for setup only. Revision 0.2 removes emergency access over Bluetooth, which was Section 14.6.3, and its mentions in Sections 9.6.3 and 14.2.2. Receiving new Wi-Fi settings after a button press (Section 9.6.4) is setup, and stays. The controller's own access point does the job of emergency access without Bluetooth (Section 14.6).
6. **Owner-installed firmware.** *Resolved in revision 0.3, as the Project Lead decided on 17 October 2026: unchanged.* Section 15.8 makes it SHOULD for products and MUST for reference designs, and is no longer provisional. Making it MUST for all products would serve owners best after support ends, but affects how manufacturers use secure boot, for example by keeping an eFuse key slot for the owner.
7. **Device Classes: firmware information.** *Resolved in revision 0.2.* Revision 0.1 carried Table 11's members in a provisional extension, `org.openaquariumproject.firmware`, until the Device Classes specification defined device attributes for them, and a fault or event for a refused or reverted update. Device Classes revision 0.2 makes them device attributes, `updateChannel`, `securityVersion`, `firmwareSigner`, `updateFeed`, `updateKeys`, `supportEnds`, and `automaticInstall`, and reports the outcome of the latest update, refused and reverted ones included, as `lastUpdate` (its Section 8.1; Appendix D, issue 15). Section 15.9 now uses them, and reads the extension only from devices of Device Classes revision 0.1.
8. **Device Protocol: carrying this Specification.** *Resolved: already answered by the Device Protocol specification.* Revision 0.1 asked how the pairing messages, invitations, and emergency access are carried, how devices are identified, and the format of the setup code's QR code. The Device Protocol carries the pairing exchange, invitations, emergency codes, and pairing lists (its Sections 10.2 to 10.6), names a device by its key ID (its Section 7.3), and defines the QR code (its Section 10.4).
9. **Device Protocol: the relay protocol.** *Resolved: already answered by the Device Protocol specification.* Its Section 15.4 defines the relay protocol: how a controller registers with a relay, how apps connect through it, and how a controller tells its apps which relays it listens on.
10. **Data Model: audit records.** *Resolved by Data Model revision 0.2: no.* Documents do not carry audit records, and a product offers to export the audit log beside history, in this Specification's format (Data Model Section 11).
11. **Automation: stopping everything.** *Resolved by Automation revision 0.2.* Section 11.8 holds outputs stopped at the precedence of interlocks. Automation revision 0.2 puts `stop-all` with interlocks in its highest layer, sends its stops with `revertAfter`, and lets it stop endpoints under local control (its Sections 17.3, 17.4.2, and 17.5).

12. **Roles.** *Resolved in revision 0.3, as the Project Lead decided on 17 October 2026.* Table 6 stands as revision 0.2 left it, and is no longer provisional: ending an emergency mode and clearing latches need the `override` role, starting step programs `full-control`, services are limited to `full-control`, and `full-control` users cannot share access. Revision 0.2 widened `start-mode` to start a mode of any kind, end one other than an emergency mode, and start or end a lighting scene, as the Project Lead decided (Automation Appendix D, issue 32), and limited `open-pairing-window` to devices, because a controller creates invitations with `manage-users` and `manage-own-credentials` (Section 11.5). Revision 0.3 adds the operations `record-maintenance` and `enter-results` (issues 29 to 31), and names the settings of a backup thermostat among the fail-safe settings that `change-fail-safe-settings` covers, as the Power Module specification asked (its Appendix E, issue 10).
13. **Numbers.** *Open: waits on the bench tests of the breadboard controller.* The window of 15 minutes, 20 attempts, 5 s for revocation, the emergency access times, the audit log's size, the confirmation time of 10 minutes, and the password rules need confirming on the breadboard controller. So do the 1 s after which `stop-all` sends `COMMAND_EMERGENCY_STOP` to a module that has not accepted its stops, and the 24 hours to which revision 0.3 limits the expiry of `stop-all` (Sections 11.8.2 and 11.8.3).
14. **Matter.** *Resolved in revision 0.3, as the Project Lead decided on 17 October 2026.* The question was how the roles map to Matter's access control. Each bridge is one user of kind `service`, whose role caps what every Matter fabric can do, whatever Matter's access control grants, and fabrics are not users of their own (Section 13.4.4). Records of what a bridge does carry `via matter`, with the fabric's index, label, and vendor ID in `details` (Section 12.1.6), in place of `local`. Turning a bridge on creates its user, so it needs `manage-users`, with `manage-consent`, as the Matter Mapping specification says (its Section 7.2.1), and has no operation of its own; opening its commissioning window and removing a fabric need `manage-users` too, because each gives or takes away access (Matter Mapping Appendix D, issue 8). Revision 0.2 already recorded each fabric added or removed (`fabric-added`, `fabric-removed`; issue 33).
15. **Certification.** *Resolved: already answered by the project's governance.* The question was whether passing the test procedures should be part of a future compatibility mark. The governance says that a compatibility program, if one is created, will require conformance testing (its Section 5), as Fail-Safe Behavior Appendix C, issue 7, notes. Which products may carry a mark belongs to the program, not to this Specification.
16. **Notifications.** *Resolved: already answered by the Alarms and Notifications specification.* The question was end-to-end encryption of notifications through third-party push services (Section 13.3.5). The Alarms and Notifications specification requires it for every push message, with Message Encryption for Web Push (its Section 17.2.3).
17. **Notification operations.** *Resolved in revision 0.3, as the Project Lead decided on 17 October 2026: unchanged.* The `Alarms and Notifications` specification uses `manage-own-credentials` for a user's own push subscriptions and quiet hours, and `manage-consent` for recipients, channels, and heartbeats, which decide what leaves the owner's network (its Table 14), and keeps them. Raised by the Alarms and Notifications specification (Appendix D, issue 8).
18. **Recording refills.** *Resolved in revision 0.3, as the Project Lead decided on 17 October 2026.* The `Water Level and Water Change` specification used `run-action` for recording that a top-off reservoir was refilled, provisionally (its Section 16). Revision 0.3 adds `record-maintenance`, which the `override` role permits, for records that change no output, such as refills, replaced reagents, and emptied waste (Table 6). Resuming or abandoning an aborted water change stays with `clear-latch`. Raised by the Water Level and Water Change specification (Appendix D, issue 12).
19. **Dosing operations.** *Resolved in revision 0.3, as the Project Lead decided on 17 October 2026.* The `Dosing` specification used `run-action` for recording that a dosing container was refilled, provisionally (its Section 11.4), and entering manual test results, which can change how much a doser gives (its Section 12.3), needed no operation. Revision 0.3 records refills with `record-maintenance`, and adds `enter-results`, which only the `full-control` role permits, so that an owner grants it on purpose, for entering test results and importing laboratory documents (Table 6). Raised by the Dosing specification (Appendix D, issue 12).
20. **Water testing operations.** *Resolved in revision 0.3, as the Project Lead decided on 17 October 2026.* The `Water Testing` specification records replaced reagents and emptied waste with `record-maintenance`, and enters manual results and imports laboratory documents with `enter-results`, from its revision 0.3 (its

Sections 9.6, 12.3, and 12.4). Starting a test on demand stays with `run-action`, applying a correction to an analyzer's results with `calibrate`, and resuming testing that it suspended with `clear-latch`. Raised by the Water Testing specification (Appendix D, issue 13).

21. **Safety functions.** *Decided by the Project Lead on 15 October 2026; resolved in revision 0.2.* Revision 0.1 said only that safety functions must not depend on a radio link alone, with no definition, no party to enforce it, and no test. Section 10.9.4 now refers to the definition of Fail-Safe Behavior Section 5.2, and to the rule of its Section 13.4 (its Appendix C, issue 14). Each safety function needs a path that uses no radio link, wired to its controller or protection inside the device, and a controller warns its owners, persistently, while one has none, rather than refusing the configuration. The test "Safety function paths" checks it (Table 14).
22. **Matter fabrics in the audit log.** *Decided by the Project Lead on 15 October 2026; resolved in revision 0.2.* Fabric administrators may open commissioning windows through Matter's own mechanism, so a fabric can be added without an owner (Matter Mapping Appendix D, issue 25). Table 8 now has the audit events `fabric-added` and `fabric-removed`, which a controller records for each fabric added to or removed from its bridge, however it happened, with the fabric's label and vendor (Section 12.1.5), which revision 0.3 gives as every record of the bridge's actions gives them (Section 12.1.6). Every owner is notified of each, as Alarms and Notifications Section 13.4.4 requires (Section 13.4.5).

The following issues arose in revision 0.2:

34. **Emergency access on a device network.** *Decided by the Project Lead on 16 October 2026; resolved in revision 0.2.* A controller with one radio can host only one Wi-Fi access point, so a controller that hosts a device network cannot also host an access point for emergency access. Revision 0.1 let it offer emergency access on its device network, but such a network is always up, and lets those who join it reach the devices on it, against Sections 14.3 and 14.6.2. The alternative was to drop the option, which would leave such a controller with only its own buttons or display for emergency access (Section 14.2.1), since Bluetooth is not used for it (issue 5). The Project Lead decided to keep the option as revision 0.2 restricts it: only for a controller that isolates the device network's clients from one another, and that accepts emergency sessions there only while Section 14.3 allows (Section 14.6.1). The reference controller may host a device network (Proposal 0001, open question 9), and an owner whose router has failed still needs to stop or run equipment from a phone. Its cost is that people who learn the device network's passphrase can join it at any time, and isolation limits what they then reach to the controller, which accepts no session from them outside an emergency. Section 14.6.1 is no longer provisional.
35. **Module Bus: an emergency stop with an end.** *Decided by the Project Lead on 16 October 2026; resolved in revision 0.2: the Module Bus needs no change.* `COMMAND_EMERGENCY_STOP` stops a whole module with one message, but carries no `revertAfter`, and, like any stop without one, cancels a pending revert (Device Classes Section 7.4.7). The Module Bus specification could have given it a `revertAfter`, in the parameter of `uavcan.node.ExecuteCommand`, as the Device Protocol's `device.stop` has one (its Section 11.4). The Project Lead decided to keep it as it is: `stop-all` sends each endpoint's stop with `revertAfter`, at the immediate priority, and sends `COMMAND_EMERGENCY_STOP` only to a module that has not accepted them within 1 s (Section 11.8.3). The stops of a module's endpoints take a few frames at the highest priority that commands use, so one message more would save little, and a parameter would change the firmware of every module. The 1 s stays provisional until bench tests confirm it (issue 14).

The following issue records the choices of revision 0.3:

36. **Choices of revision 0.3.** *Resolved in revision 0.3: the Project Lead accepted each choice on 18 October 2026.* Revision 0.3 made these choices where the decisions it applies left them open:
 - `stop-all` lasts at most 24 hours (provisional), so that its stops end by themselves within a day if the controller fails, and an owner who needs longer starts it again (Section 11.8.2);
 - `record-maintenance` needs the `override` role, so that a person looking after the system can record a refill, and `enter-results` the `full-control` role, so that only a person trusted to change how the system works can enter results that steer dosing (Table 6);
 - opening a Matter bridge's commissioning window, and removing a fabric from it, need `manage-users`, as turning the bridge on does, because each gives or takes away access (Section 13.4.4); and

- an app relays a web interface as HTTP over a connection authenticated with its identity key, so that the person needs no password, and guards what it listens on as Section 10.4.6 requires [OAP-DP Section 15.5].

PRE-DRAFT