

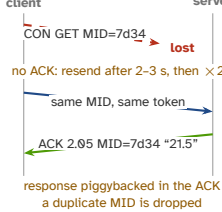
In one line: CoAP (RFC 7252) is **REST over UDP**: GET/POST/PUT/DELETE and HTTP-like codes in a **4-byte binary header**, reliability by **confirmable** messages with exponential backoff, push by **Observe**, big bodies by **block-wise**, security by **DTLS** or **OSCORE**. **LwM2M** (OMA SpecWorks) is **device management on top of CoAP**: a standard **object/instance/resource** tree (`/3/0/3` = firmware version) plus bootstrap, registration, management and reporting interfaces.

CoAP message — 4-byte fixed header

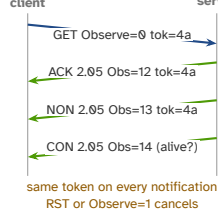
Ver	T	TKL	Code c..dd	Message ID (16 bit)
Token (0–8 bytes, length = TKL)				
Options (delta-encoded): Uri-Path, Observe, Block2...				
0xFF		Payload		

Ver = 1 · T: 0 CON, 1 NON, 2 ACK, 3 RST
 Code: 0.01 GET 0.02 POST 0.03 PUT 0.04 DELETE
 2.01 Created 2.04 Changed 2.05 Content 4.04 5.00

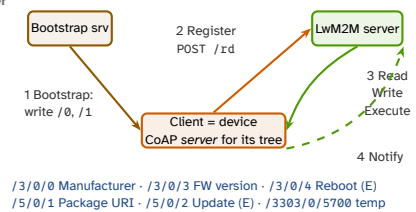
CON retransmit



Observe (RFC 7641)



LwM2M — who talks to whom



CoAP — how it works

- UDP port **5683**, DTLS **5684**; TCP/TLS/WebSocket binding exists (RFC 8323) but UDP is the design point: no connection state, no head-of-line blocking.
- Two layers**: *message* layer (type + 16-bit **Message ID**: dedup, ACK matching) and *request/response* layer (**Token**, 0–8 bytes: matches responses and notifications to a request). Response = **piggybacked** in the ACK, or **separate** (empty ACK now, CON response later).
- Reliability**: CON waits for ACK; timeout random in 2–3 s (ACK_TIMEOUT 2 s × factor 1.5), doubled each time, **4** retransmits max. NON = fire-and-forget telemetry. RST = “I can’t handle this”.
- Options** are delta-encoded numbers: Uri-Path, Uri-Query, Content-Format (0 text, 50 JSON, 60 CBOR), Max-Age, ETag.
- Observe**: best-effort push of the *latest* state (intermediate states may be skipped), sequence numbers reorder, at least one CON per 24 h checks the observer is alive.
- Block-wise** (RFC 7959): Block2 = response body, Block1 = request body; NUM + More + size $2^{5ZX+4} = 16...1024$ B — firmware, fits 6LoWPAN MTUs.
- Discovery**: GET `/well-known/core` → CoRE Link Format (`</temp>;rt="temperature";if="sensor"`).
- Security**: DTLS with PSK, raw public key or X.509 (handshake RAM + round trips hurt); **OSCORE** (RFC 8613) protects the CoAP message itself end-to-end, through proxies.

CoAP vs MQTT vs HTTP

	CoAP	MQTT	HTTP/1.1
Transport Pattern	UDP (+DTLS) request/response + Observe	TCP (+TLS) pub/sub via broker	TCP (+TLS) request/response
Topology	client ↔ device, no broker	star around broker	client → server
Header Sleepy / NAT	4 B binary no session; NAT binding expires → server can't reach device	2 B min keepalive holds TCP	text, 100s of B poll
Best at	RESTful device mgmt, NB-IoT, 6LoWPAN	fan-out telemetry, many consumers	phones, web, cloud APIs

LwM2M — the model

- Path `/Object/Instance/Resource`: **0** Security · **1** Server · **2** Access Control · **3** Device · **4** Connectivity Monitoring · **5** Firmware Update · **6** Location; IPSO objects (3303 Temperature). Standard IDs = interoperable servers.
- Resources are R, W or E (execute: POST `/3/0/4` reboots). Formats: TLV, SenML JSON/CBOR, plain text.
- Roles are inverted: the device is a CoAP *client* to register, and a CoAP *server* for its object tree.

The four interfaces

Bootstrap	factory-known bootstrap server writes <code>/0</code> (server URI + DTLS keys) and <code>/1</code> (lifetime, binding)
Registration	POST <code>/rd?rep=<name>&lt;lt=86400 + link list (</3/0>,</5/0>)</code> → 2.01 + Location; Update before <code>lt</code> expires; DELETE = de-register
Device mgmt & service enablement	Read (GET) · Write (PUT/POST) · Execute (POST) · Create · Delete · Discover · Write-Attributes
Information reporting	Observe / Notify / Cancel; <code>pmIn/pmax</code> attributes bound the notify rate

Firmware update with Object 5

Write `/5/0/1` Package URI → device downloads (Block2) → State `/5/0/3`: 0 Idle → 1 Downloading → 2 Downloaded → server Executes `/5/0/2` → 3 Updating → reboot → Update Result `/5/0/5`. **Queue mode**: a sleeping device's requests wait until its next Update.

Interview traps

- ! “CoAP runs over TCP” — UDP is the whole point.
- ! Message ID ≠ Token: dedup/ACK vs request matching.
- ! Observe is not a reliable stream — design for skipped notifications.
- ! “LwM2M replaces MQTT” — management vs data; they coexist.

Remember

CON waits, NON flies · MID for the message, token for the request · Observe = latest · LwM2M = `/object/instance/resource`.

Likely questions

- Constrained sensor: CoAP or MQTT? — CoAP for sleepy UDP/NB-IoT and REST mgmt; MQTT for fan-out through a broker.
- Big payload over CoAP? — block-wise, never IP fragmentation.
- Where does the phone fit? — behind a gateway/proxy speaking HTTP/MQTT.