Metering protocols: DLMS/COSEM, OBIS, Modbus, MQTT (and NFE CHINT register map)

> Traceable explainer: how metering data actually moves, where DLMS/COSEM, OBIS, Modbus, MQTT and HTTP each sit, and how NFE's live CHINT registers map to the OBIS standard vocabulary. Compiled 2026-08-10. Companion to "NFE metering strategy + Glen Street-EMS".

The layer cake (mental model)

Networking is layers. Each solves one problem and talks only to the layer above and below.

Layer	Job	Examples
Link	move bits over one physical medium	RS485, Ethernet, WiFi
Network (IP)	address and route between any machines	IP
Transport (TCP)	reliable, ordered byte-stream between two programs	TCP, UDP
Application	what the two programs actually say	HTTP, MQTT, Modbus, DLMS

Key point: **MQTT, HTTPS and DLMS/COSEM are all application-layer.** MQTT and HTTPS run on top of TCP (transport) on top of IP (network). MQTT is NOT a transport or network protocol.

> One clean way to hold it all: IP gets you to the right building. TCP is registered mail that guarantees the whole message. Modbus/DLMS is the local interview with one meter (DLMS adds a shared vocabulary and an ID badge). OBIS/COSEM/SunSpec is that shared vocabulary. MQTT is the courier network carrying notes across the village and orders back. HTTP is a different courier

> that only does "go fetch one thing and return".

- MQTT: application, over TCP, port 1883 (8883 with TLS)
- HTTPS: HTTP + TLS over TCP, port 443
- DLMS/COSEM: application; rides serial (HDLC) OR TCP/IP directly
- Modbus RTU (NFE's meter to Pi link): application, straight over RS485 serial, no IP at all (why it is the simplest thing on site)

Three separate jobs, often confused

- **Vocabulary** (what a value means): OBIS / COSEM for meters, SunSpec for inverters.
- **Local interrogation** (talk to one device): Modbus or DLMS.
- **Messaging** (move data between many nodes, up and down): MQTT or HTTP.

You put COSEM-modelled data inside an MQTT message. They are not competitors: MQTT is the envelope, COSEM is the language on the page.

MQTT vs HTTPS

- HTTPS is request/response, pull: client asks, server answers, one round trip. The server cannot easily push to a device.
- MQTT is publish/subscribe through a broker, push: a device holds one persistent TCP socket to a broker and publishes to a topic; subscribers get pushed the message instantly. Tiny 2-byte header, QoS levels, retained last value, last-will (detect a dead node), pushes both ways (telemetry up, commands down), survives flaky low-bandwidth links.
- The Street-EMS village dashboard uses MQTT: it pushes "TURN OFF SSR0" down to a specific cabinet on topic `openami/StreetPoleEMS_<id>/cmd` while readings stream up.

MQTT over the internet? Yes, it is just TCP. Devices make an outbound connection to a broker, so no inbound port-forwarding through the site modem, ideal for field gear on cellular.

Why not DLMS/COSEM over HTTPS? It buys nothing: DLMS already has its own TCP transport and its own message-level crypto (AES-128-GCM). And DLMS is a point-to-point session with one meter, not a fan-in/out telemetry bus. So the natural split is DLMS or Modbus to talk to the meter locally, MQTT to move the results around.

DLMS/COSEM, unpacked

- **COSEM** (Companion Specification for Energy Metering) is the data model: every value is an object of an interface class (Register, Profile Generic for load profiles, Clock, Disconnect Control for the relay, etc.).
- **OBIS** (OBject Identification System) names each object with a six-group code A-B:C.D.E.F. Example: `1-0:1.8.0` = total active energy imported (what you bill).
- **DLMS** (Device Language Message Specification) is the protocol: GET an attribute, SET an attribute, ACTION a method, inside an authenticated, encrypted association.
- Standardised as IEC 62056. Certified by the DLMS User Association.

The distinction: Modbus register vs COSEM "register"

A meter does NOT contain Modbus registers and COSEM registers side by side. They are two alternative languages for the same physical measurements. A given meter speaks one (or, on high-end dual-interface meters, both via separate ports).

- **Modbus register** = a numbered 16-bit memory slot at an address (e.g. `0x2004`). Dumb storage. It is meaningless without the vendor manual telling you "0x2004 is active power". The meaning lives in a PDF, not in the data.
- **COSEM Register (object)** = a self-describing object that carries its own unit and scaler, addressed by a global OBIS code that means the same thing on every compliant meter. (COSEM has an interface class literally named "Register", class_id 3 - hence the word collision. It is an object, not a memory slot.)

NFE's CHINT DDSU666 / DTSU666 are **Modbus-only** meters: inside them there are only Modbus registers, no COSEM, no OBIS. The OBIS column below is a translation NFE applies in software (in the logger / EMS / OpenAMI layer); the meter itself never knows about OBIS. The value of doing it: the meaning stops living in the CHINT manual and becomes the industry-standard, self-describing vocabulary.

NFE CHINT register to OBIS map

Single-phase DDSU666 (tenant meters, one phase). Source: `nfe-modbus-energy-logger/src/meter_reader.py` and the OpenEMS `Meter.Chint.DDSU666` driver.

Quantity	Modbus register (float)	OBIS code	Meaning
Voltage	0x2000	1-0:32.7.0	Instantaneous voltage L1
Current	0x2002	1-0:31.7.0	Instantaneous current L1
Active power	0x2004	1-0:1.7.0	Instantaneous active power (import +)

Quantity	Modbus register (float)	OBIS code	Meaning
Power factor	0x200A	1-0:13.7.0	Power factor
Frequency	0x200E	1-0:14.7.0	Frequency
Import energy	0x4000	1-0:1.8.0	Total active energy imported (billing)
Export energy (invert cfg)	0x4000	1-0:2.8.0	Total active energy exported

Three-phase DTSU666 (aggregate meter_100):

Quantity	Modbus register(s) (float)	OBIS code(s)	Meaning
Voltage L1/L2/L3	0x2006 (x3)	32.7.0 / 52.7.0 / 72.7.0	Instantaneous phase voltages
Current L1/L2/L3	0x200C (x3)	31.7.0 / 51.7.0 / 71.7.0	Instantaneous phase currents
Active power total + L1/L2/L3	0x2012 (x4)	1.7.0 / 21.7.0 / 41.7.0 / 61.7.0	Instantaneous active power
Power factor	0x202A	13.7.0	Power factor total
Frequency	0x2044	14.7.0	Frequency
Import energy	0x4000	1-0:1.8.0	Total active energy imported

Note: the DDSU666 driver reads active power as already scaled kW; the DTSU666 needs per-manual scaling (Table 11: V/10, I x0.001, P x0.0001, PF x0.001, F x0.01). Minor code note: the logger's DTSU `REGISTERS` dict lists PF at 0x2020 but `read()` polls 0x202A; 0x202A is the one in use.

Why two OpenEMS drivers (meter and inverter)

Two physical devices with different proprietary Modbus maps means two drivers:

- CHINT meter (`MeterChintDdsu666Impl`) reads 0x2000+; implements the `ElectricityMeter` Nature.
- SRNE inverter (`SrneEssImpl`) reads a totally different map at 0x0100 (SoC), 0x0101 (voltage), 0x0102 (current); implements the `SymmetricEss` / `HybridEss` Natures.

OpenEMS "Natures" (ElectricityMeter, SymmetricEss) are abstract interfaces both drivers map into, so the rest of OpenEMS sees a uniform API regardless of vendor. That is the local equivalent of what OBIS/COSEM and SunSpec do industry-wide.

The real prize of the standards is not one driver for meter plus inverter. It is one driver per device TYPE that works across every compliant vendor: any DLMS meter uses the same meter driver, any SunSpec inverter the same inverter driver. That removes the per-vendor Modbus-map work NFE did for both the CHINT and the SRNE. OpenEMS Natures give that uniformity on the edge (after the driver); DLMS and SunSpec give it at the wire (removing the driver work).

OBIS vs profiles (IDIS / OpenAMI): naming is not interoperability

OBIS names things, it does not force a meter to implement or behave the same way. DLMS/COSEM is a large, flexible toolkit and leaves most choices to the manufacturer: which OBIS objects are actually present, which data types and scalers, which security suite, which communication profile, which optional events. So two meters can both be DLMS/COSEM compliant and both use OBIS and still not interoperate out of the box. Compliant is not the same as interoperable.

IDIS (Interoperable Device Interface Specifications) is an industry association profile that sits on top of DLMS/COSEM/OBIS and removes the optionality: it mandates exactly which OBIS objects must be present and how they behave, the data types, security and events, a fixed communication profile, and an interoperability certification (conformance plus multi-vendor testing). Packages by transport: Package 1 (basic, local/optical + core residential objects), Package 2 (power-line comms PRIME/G3 + data concentrators), Package 3 (cellular/IP). Any two IDIS-certified devices genuinely interchange.

Mental model: DLMS/COSEM is the grammar and a giant dictionary with optional dialects; OBIS gives every word one fixed meaning; IDIS is a mandated phrasebook plus accent everyone certifies to, so a whole conversation is portable, not just the words.

Why it matters here: even if two meters both spoke OBIS, if they exposed different object sets over different links you would still write per-vendor glue (the same thing that produced two OpenEMS drivers). A profile is what makes "buy any certified meter, one driver, plug and play" real. IDIS is the heavyweight, utility-grade version of exactly what OpenAMI aims to be for off-grid villages: a mandated, open, lightweight village-metering profile on top of the vocabulary. NFE does not need full IDIS, but it does need a profile, and that is OpenAMI's whole purpose.

How OpenAMI sits on top

OpenAMI (IEEE Smart Village) is the actionable, utility-facing telemetry layer above a village feeder. It does not reinvent meter semantics: it composes OBIS/DLMS/COSEM (meters), SunSpec (inverters/DER), MQTT (transport), IEEE 2030.5 and 1547 (DER control). Its value-add is two-way, village-edge, policy-driven control and a billing cache, run cloudlessly and cheaply.

Sources

Element room "Smart Village Energy Interest Group"; isv.wiki; `nfe-modbus-energy-logger/src/meter_reader.py`; OpenEMS drivers `io.openems.edge.meter.chint.ddsu666` and `io.openems.edge.ess.srne`; IEC 62056 (DLMS/COSEM); DLMS User Association Blue Book (OBIS).

Revision #3
Created 2026-08-09 20:35:52 UTC by hillary.arinda
Updated 2026-08-09 21:26:31 UTC by hillary.arinda