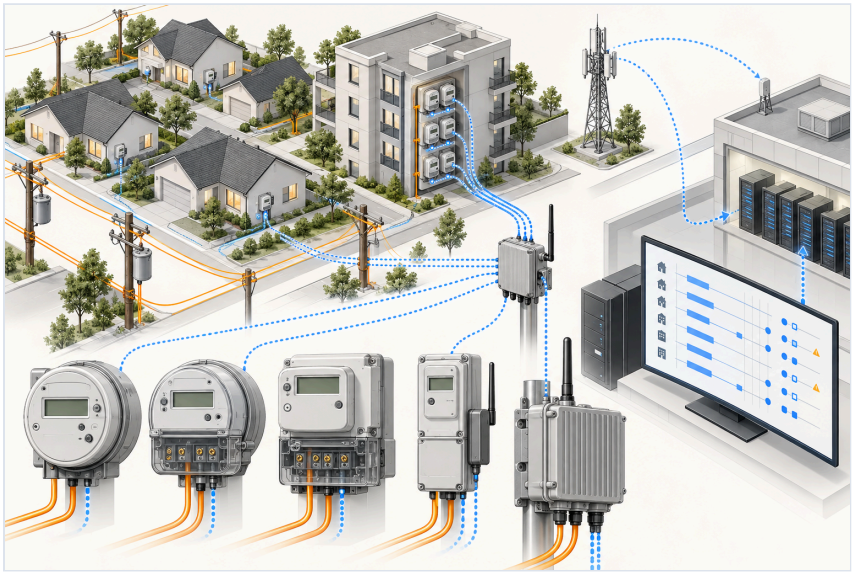
AETHERONE / AETHERHES

AetherHES Catalogue

Following a reading, a command and a communication fault through the Head End System (HES) · Document edition 2026.10



CONCEPT ILLUSTRATION; NOT A CUSTOMER INSTALLATION OR RUNTIME EVIDENCE.

Concept illustration of an AMI acquisition path: smart meters in houses and an apartment-block meter cabinet, meters of several designs, pole-mounted data concentrators with antennas and a cellular tower, with blue communication paths leading to head-end server racks and a collection monitor. It shows the path AetherHES serves and is not a customer installation or a record of operation.

Between the meter and the bill

A kilowatt-hour starts as a measurement inside a meter and ends as a line on a bill, a work order or a decision. Between the two sits a communication layer that has to answer four questions every time: which device spoke, over which session, which bytes arrived, and what was done with them. In AetherONE that layer is AetherHES, the Head End System.

Look at the street in the illustration. Meters hang on house walls and in an apartment-block cabinet. Data concentrators on poles gather them, a cellular tower carries the traffic, and a room of servers waits at the far end. HES is the system in that room, and it is responsible for everything that happens between the first frame and the moment a reading is handed on. It keeps the communication identity of each device and the protocol sessions it speaks over. It runs reads and collection, records what became of every command, follows link telemetry and device events, delivers firmware and configuration, and keeps the raw communication record (Raw Chronicle) in a private store.

Other products see only what a versioned contract allows: typed fields, references, hashes and status. Measurement truth stays with AetherMDMS, spatial truth with AetherGIS, financial billing with AetherRevenue and field work with AetherDispatch.

An integration engineer follows one reading

The engineer holds one captured meter frame and one question: what happened to it after it arrived?

A local TCP adapter accepts the frame as newline-delimited JSON from a deterministic simulator. The session is stored as open or closed in an HES-owned SQLite WAL database, bound to the device it was observed from, and it survives a restart. The frame goes into the transactional Raw Chronicle, and a durable outbox sits behind the same adapter session, so a decoded event and its raw record stay together in HES-owned storage.

The decoded event follows a versioned contract. It carries a computed raw-frame hash, a computed payload hash and references, and its payload is a bounded kW reading. A separate broker spool publishes the event with replay-safe delivery and records the acknowledgement that later comes back from MDMS. The engineer can follow one reading through its event, correlation, causation and hash links, and the raw bytes never leave HES.

1 One decoded measurement: event, contract hes.measurement.decoded.v1, schema 1.0.0 and device

Event	Contract	Schema	Device
EVT-HES-99141-DECODED	hes.measurement.decoded.v1	1.0.0	KEP88091044

2 Collection plans: name, kind, schedule and data set

Plan	Name	Kind	Schedule	Data set
CP-1001	LP 15-min all commissioned	Scheduled Read	*/15 * * * *	Load Profile 1
CP-1002	Daily frozen 00:10	Scheduled Read	10 0 * * *	Daily Frozen
CP-1003	Monthly billing read	Billing Read	0 1 1 * *	Billing Registers
CP-1004	Seoul-East gap backfill	Backfill	manual	Load Profile 1

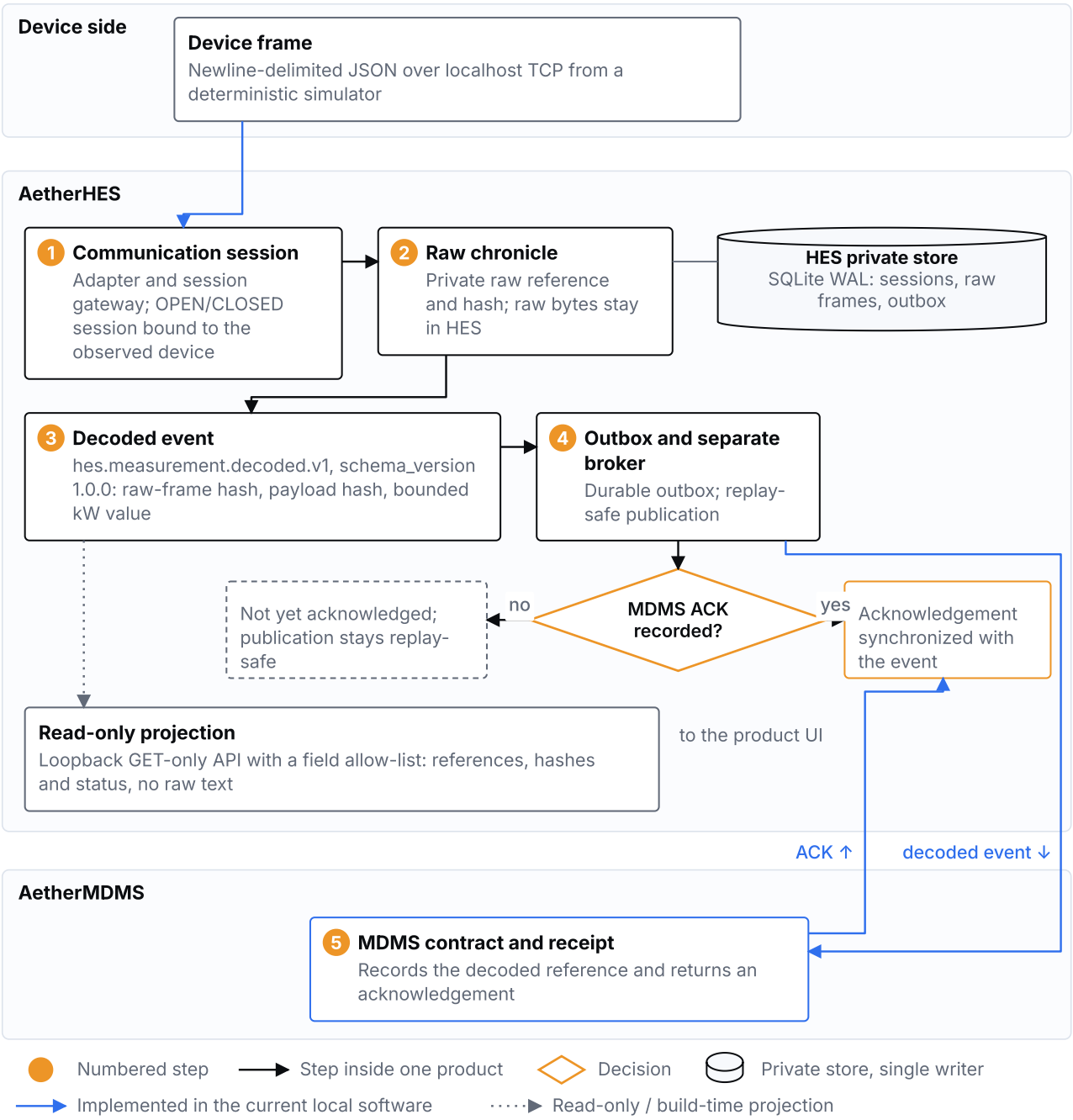
PRODUCT SCREEN · DEMONSTRATION DATA Evaluation interface with demonstration data: the Collection view. The first excerpt shows one decoded measurement with its event, contract, schema version and device. The second lists collection plans with name, kind, schedule and data set. Plan names, schedules and device identifiers are synthetic.

Read the figure from the top. The first excerpt names a single decoded event and the contract it follows. Below it, collection plans pair a kind with a schedule and a data set. In the evaluation edition the Collection page opens on local live evidence by default: decoded reads, raw-frame references and hashes, outbox state, broker delivery and the MDMS acknowledgement. The demonstration plans appear only when the engineer explicitly selects demonstration mode, and a failed read never falls back to them. The page reads through a read-only API whose field allow-list excludes raw text, stored event JSON and receipt JSON.

AetherHES: one reading, from session to MDMS receipt

DESIGN WORKFLOW

Current local slice: one host, one adapter, a deterministic frame.



DESIGN WORKFLOW Design workflow: communication session, private raw reference and hash, decoded event, separate broker, MDMS contract and receipt. This is a conceptual design diagram of the intended path of one reading and does not describe a production integration.

The product is designed around this path: one session, one private raw reference, one decoded event and one receipt, linked end to end.

An operations engineer separates acknowledgement from outcome

A command sits on the screen, and the engineer has to decide whether it is finished.

The contract defines eight stages: requested, authorized, queued, dispatched, transport-acknowledged, device-acknowledged, read-back verified and business outcome confirmed. The whole chain is built around the distance between the sixth stage and the eighth. A device that acknowledged a command has not necessarily changed state, and a changed state is not yet a business result. Device acknowledgement is therefore never a final state.

The command component of the evaluation edition is a resident single-instance pump that reads requests from one spool and appends ordered status evidence to another. It consumes each request idempotently, binds the target to the session that was actually observed, and records transport acknowledgement, device acknowledgement and a matching read-back. After a process or SQLite restart it carries on without repeating the attempt. HES reports execution evidence and nothing more. In the test-mode token flow, AetherMDMS alone projects the business completion once the read-back matches, and financial authority stays with AetherRevenue. The product calls this surface the Command Execution Queue because it is the device-facing execution queue of HES, not utility field dispatch.

1 Command queue: command, kind and target

Command	Kind	Target
CMD-50001	Relay Disconnect	MTR-000101
CMD-50002	Time Sync	COHORT: Seoul-North commissioned
CMD-50003	TOU Write	MTR-000105
CMD-50004	Relay Connect	MTR-000104
CMD-50005	Demand Reset	COHORT: Billing cycle A

PRODUCT SCREEN · DEMONSTRATION DATA Evaluation interface with demonstration data: the command queue, listing synthetic commands with their kind and target. Commands and targets are synthetic, and no command or relay was operated.

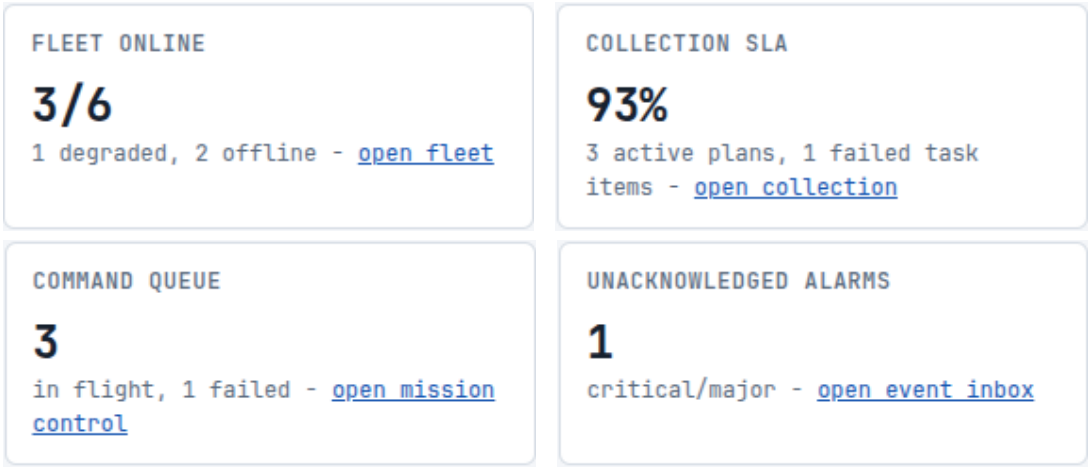
The queue lists each command with its kind and target, the way an operator sees them before opening one. In the design, opening a command shows its receipt moving through the eight stages described above, so a command that stopped at device acknowledgement reads as unfinished rather than done. In this edition the screen is a demonstration interface on synthetic commands, and its controls are demonstration controls.

A communication operator walks the fleet, the links and the firmware

The operator looks after meters, DCU and SMGW gateways, their uplinks, firmware campaigns and device events, and wants to see how a degraded link or an unacknowledged alarm would surface.

The Operations Center puts fleet status, collection SLA, the command queue, unacknowledged alarms, network degradation and firmware campaigns on one page, and each tile leads to the module that owns it. Beneath the tiles, an adapter health table and a gateway uplink table show where traffic comes from and which devices hang behind each concentrator. Topology and link quality, the firmware repository, campaign progress, configuration drift, the event stream, the alarm inbox and a raw-frame viewer complete the working surface.

1 Fleet, collection, command queue and alarm summary



2 Gateway uplinks: DCU and SMGW, uplink type and attached devices

Gateway uplinks

Gateway	Kind	Uplink	Children	State
DCU-0007	DCU	Cellular	3	ONLINE
DCU-0009	DCU	Cellular	2	DEGRADED
SGW-0021	SMGW	Ethernet	1	ONLINE

PRODUCT SCREEN · DEMONSTRATION DATA Evaluation interface with demonstration data: the Operations Center. The first excerpt summarizes fleet, collection, command queue and alarms. The second lists gateway uplinks, with DCU and SMGW gateways, uplink type and attached devices. All counts and percentages are synthetic and are not measured performance.

These views show the operating picture the product is designed to give an operator, and they are built from synthetic demonstration data. The network map is a projection, because AetherGIS owns spatial truth. Firmware storage and signing, staged rollout with post-upgrade read-back, and event ingestion with deduplication and correlation belong to the approved scope and are not part of this edition.

Availability and evaluation edition

AetherHES has an approved design scope, and this edition makes part of it available for controlled local evaluation: single-host components built around a deterministic simulator, namely the session gateway, the Raw Chronicle with decoded events and a broker spool, read-only views and a test-mode command flow. The operations-center views are demonstration interfaces. Production protocol packs for specific meter and vendor fleets, certification, operator authentication and approval, high availability and live field integration are outside this edition, and the command flow is a test-mode token delivery rather than production STS or dispatch to vendor devices. No throughput, latency or capacity figure is published, and the Raw Chronicle is a transactional local store that is not presented as tamper-proof.

Evaluating locally uses the private web package (version 0.1.0, not a commercial release), and the command flow needs an explicit shared broker path with paired request and status spools. The datasheet lists the runtime prerequisites.

To request the datasheet or speak with the ARCHAI team, use the download and contact links that accompany this catalogue.