AETHERONE / AETHERREVENUE

AetherRevenue

Catalogue

Revenue operations for AMI utilities

A smart meter on a house wall reports its consumption interval by interval, and later a customer opens a bill. Between the two sit tariffs that change on a date, a population of accounts that must be billed together, payments that arrive by several routes, and prepaid customers whose credit is spent a day at a time. AetherRevenue is the component of AetherONE designed to handle the financial half of that journey: the tariff, the financial bill, the payment, the reconciliation, and the prepaid sale.



CONCEPT ILLUSTRATION; NOT A CUSTOMER INSTALLATION OR RUNTIME EVIDENCE.

Concept illustration of the AMI revenue chain that AetherRevenue is designed for, from a meter at a premise to a ledger line under review. It is not a customer site or a record of real accounts.

Where billing starts and stops

In the approved design, the work begins with usage that someone else has already prepared. AetherMDMS validates the meter data and hands over Bill Determinants, the technical input to billing. AetherRevenue takes over from that handoff. It is responsible for applying the tariff version that was in force, building the financial bill, and following what happens to it afterwards: payment, allocation, arrears, accounting entries. A Bill Determinant is never a bill. The financial bill is built in AetherRevenue.

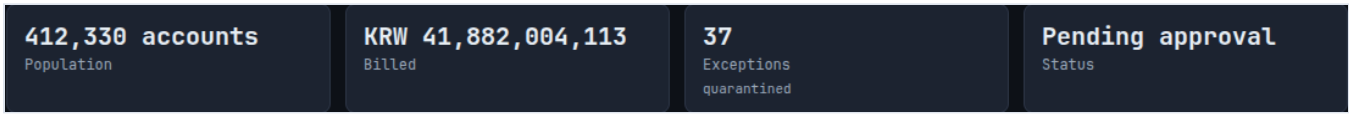
Account vocabulary needs the same care. AetherCIS owns the customer account, including its identity and lifecycle, as well as the premise, the Service Point and the agreement. AetherRevenue owns the financial work that refers to accounts: tariff, billing, payment, debt, settlement and prepayment. Its contracts carry the account references that billing and payment use. Under the same design each product keeps its own private store as its only writer, and the products meet through versioned contracts and projections, never through a shared database.

The billing analyst: one run, thirty-seven exceptions

For a billing analyst the day starts with the tariff. The design gives AetherRevenue the tariff versions, with effective periods, so that an analyst can design and simulate them against golden cases and see which version is assigned to which billing account from which date. In the demonstration, golden cases for residential, commercial and industrial tariffs each set an expected amount beside the calculated one. A tariff that does not reproduce its golden cases is a problem to find before a bill run, not afterwards.

The bill run is where those versions meet the whole population. The demonstration run in the figure covers 412,330 accounts and KRW 41,882,004,113 of billed amount. Thirty-seven accounts are held back as exceptions, and the run is waiting for approval. The reasons are specific and different in kind. One account shows a negative usage delta against its register projection. One is missing the interval projection for 3 August 2026. One has two tariffs overlapping on the same effective date. The first two point at the usage input. The third points at tariff assignment, which belongs to Revenue.

1 One bill run: account population, billed amount, exceptions and approval status



2 Quarantined exceptions with their reasons

Exceptions (quarantined)

ACCOUNT	REASON
BA-1003	Negative usage delta vs register projection
BA-1044	Missing interval projection for 2026-08-03
BA-1099	Tariff overlap on effective date

3 Approval trail of the run; the engine version label is demonstration data, not a product release

Approval trail

1. 2026-08-06T09:12:00+09:00 **Population locked** 412,330 accounts, 37 exceptions quarantined
2. 2026-08-06T16:40:00+09:00 **Rating complete** Deterministic engine R-2026.8
3. — **Approval** awaiting revenue operations approver

PRODUCT SCREEN · DEMONSTRATION DATA Evaluation interface with demonstration data. Three excerpts of one synthetic bill run: the population, billed amount, exceptions and status; three quarantined exceptions with their reasons; and the approval trail. Accounts, KRW amounts and times are examples.

The approval trail keeps the order of events in plain view. The population was locked at 09:12 on 6 August 2026 (UTC+09:00), rating finished at 16:40, and the last line shows approval still waiting for a revenue operations approver. An approver who reads it with the exception reasons beside it can see what was excluded and what the run is about to release. Each bill in the demonstration can be opened with its calculation trace, which lists steps such as the usage projection, the tier split and the tariff version applied, each with its input and output. When a bill needs to change, the adjustment view carries the rebill, credit note or waiver together with its reason and status.

The reconciliation operator: from payment to a ledger that closes

A payment is not a bill, and an allocation is not a payment. The design keeps them apart because a finance operator needs each as separate evidence. A payment arrives by channel, a bank transfer or a card gateway. It is allocated to bills, or it stays unallocated until someone decides where it belongs. In the demonstration, a bank transfer of KRW 1,000,000 is received and not yet allocated, which is exactly the kind of item an operator is meant to pick up.

Reconciliation then compares the utility's records with the outside world. The demonstration summary sets 1,204 bank statement lines and 8,431 gateway settlements against 9,512 matches made automatically and 123 left unmatched. The unmatched items listed carry hints such as a possible partial payment, a missing merchant fee split or an unknown reference text. Beyond that, the debt and collections views group arrears into aging buckets and follow each collection case through its stages. Requests to disconnect or reconnect a service pass through an approval step, and in this edition they are approvals to look at, with no field action behind them.

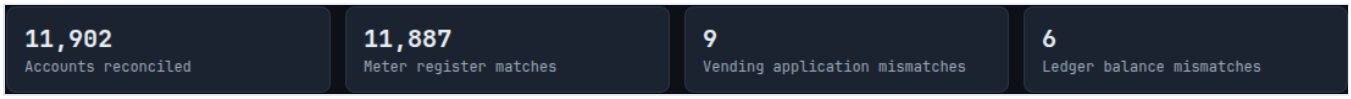
The design has the revenue subledger record journals that point back to the bill run or payment they came from. Export batches then go towards AetherFinance, a separate product with its own data. The export view lists each batch and whether it has been acknowledged, and AetherRevenue is not designed to read or write what AetherFinance keeps.

The prepayment operator: a vend that has to agree in three places

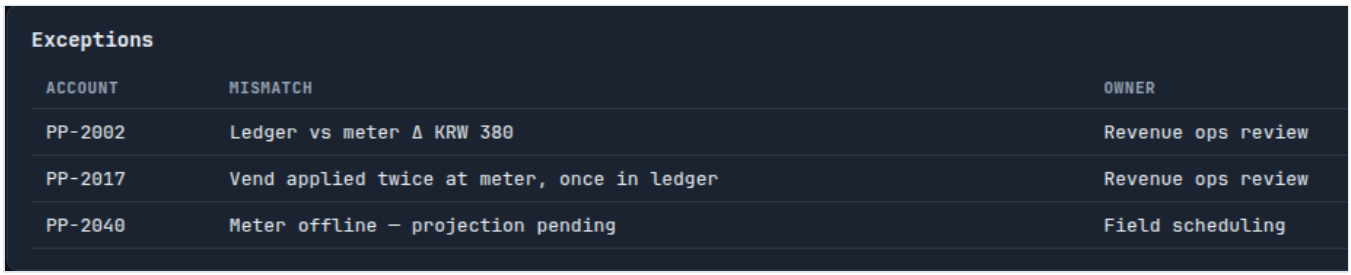
For a prepaid customer the money comes first. The customer pays a merchant, the sale is captured at a terminal, a token reference is issued, the meter applies the credit, and the account ledger is credited. Each step happens in a different place, and the results have to agree. The design treats sale and payment, delivery, meter confirmation and reconciliation as separate stages, and it keeps the merchant side as well: the merchant, its outlets and terminals, credit limit, commission, and settlement periods with gross, commission and net amounts.

On the account side, the balance ledger is append-only. Vend credits and daily energy debits are entries with balances after each one, and a meter reconciliation view compares the ledger balance with the meter's credit register projection and checks that the last vend was applied. A vending transaction can also be followed through its lifecycle, including the case where a customer dispute opens a reversal review.

1 Three-way prepayment reconciliation: meter registers, account ledgers and vending



2 Mismatches with the team responsible for each

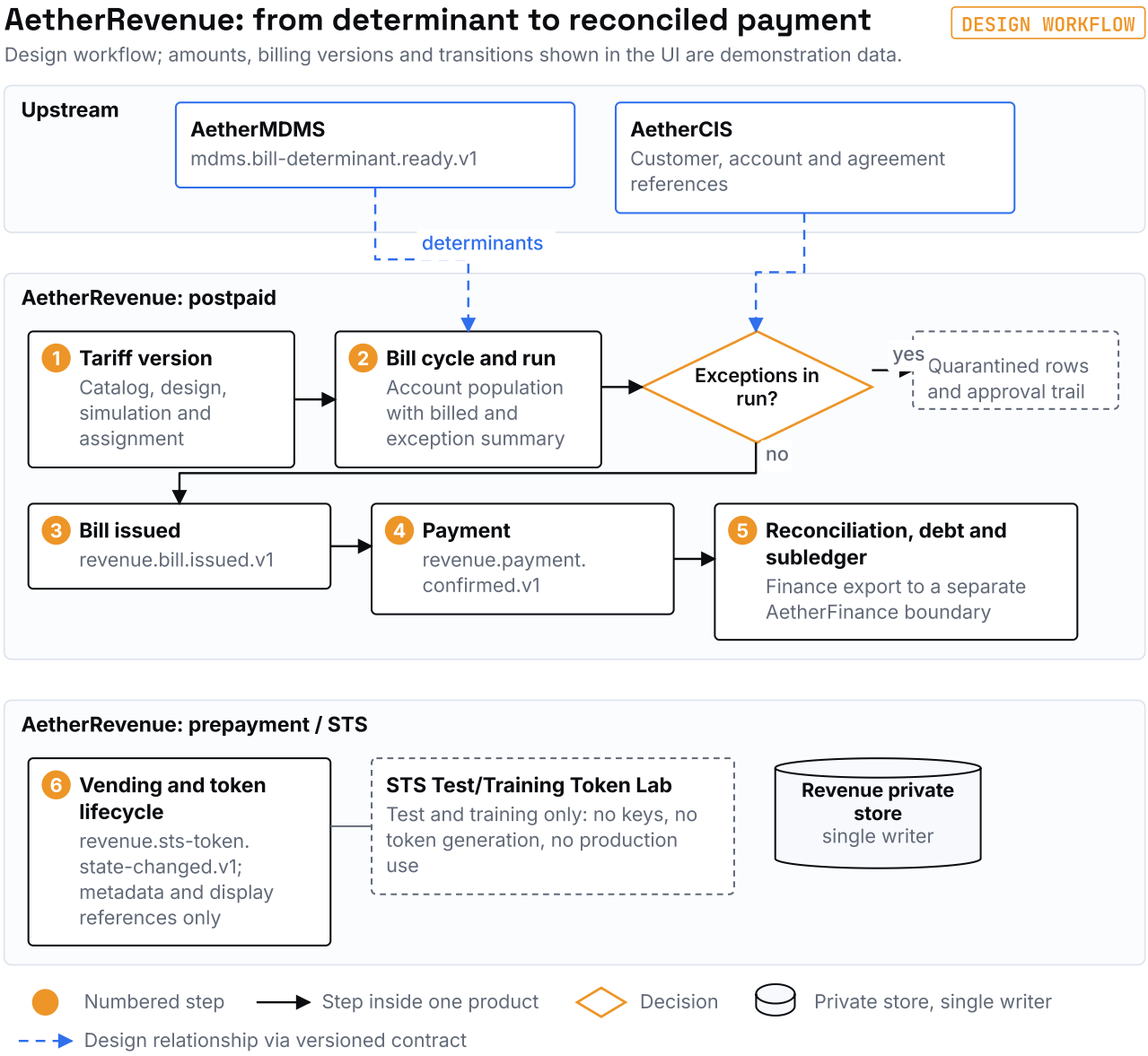


PRODUCT SCREEN · DEMONSTRATION DATA Evaluation interface with demonstration data. A synthetic prepayment reconciliation: accounts reconciled and the counts of matches and mismatches, then three mismatches, each with its responsible team. All values are examples.

The demonstration reconciliation covers 11,902 accounts. Of these, 11,887 match their meter register. Nine vending applications and six ledger balances do not agree. The listed mismatches show why the question needs three sources. One ledger differs from its meter by KRW 380. One vend was applied twice at the meter and recorded once in the ledger. One meter is offline, so its projection is still pending. The first two go to revenue operations review, and the offline meter goes to field scheduling.

STS appears here as metadata. Token lifecycle, SGC, KRN, KEN and TID metadata, a projection of security-module status and a key-change campaign view are display references. They contain no token value, no key and no module control. A separate training lab uses fixed public vectors inside a TEST SECURITY DOMAIN, so that people can learn decoder and lifecycle steps without touching anything real.

How the pieces connect



DESIGN WORKFLOW Design workflow of AetherRevenue. It shows the intended order of responsibility from Bill Determinant to reconciled payment and the separate prepayment branch. It describes design, and no integration shown in it is claimed to be running.

An integration team can read the diagram as a map of who writes what. In the design, AetherMDMS announces a Bill Determinant as ready, and AetherRevenue consumes it. AetherRevenue states when a bill is issued, when a payment is confirmed and when a token intent changes state, and it carries a token fingerprint rather than the token. AetherCIS is the documented source of versioned customer and service references. Four versioned contracts at schema version 1.0.0 describe those exchanges. Bank links, gateway links and meter confirmation are decisions of a particular project, and the plan for them belongs with that project's integration team.

Scope and availability

The responsibilities described above are the approved design. What this edition provides is an evaluation interface for the same areas, built to show reviewers the screens, the objects and the responsibilities. It opens without a data source. Choosing the demonstration mode loads fixed synthetic records, including every account, KRW amount and time in the figures. The amounts and approvals are examples and not real payments, bills, balances or financial acceptance.

The controls for approving a run, vending, rebilling, refunding, reissuing and running a campaign tell the user that nothing was executed, and the interface has no connected API, authentication or persistence. Bills are financial bills in a demonstration and carry no tax-invoice claim. This edition makes no claim of STS certification, a live security module, protocol compatibility, capacity or customer deployment. A utility would need to settle its tariff rules, payment channels, finance integration, security and the MDMS and CIS contract versions before the design could run on real accounts. Document edition 2026.10 identifies this catalogue and is not a commercial release.