AETHERONE / AETHERREVENUE

AetherRevenue

Datasheet

Revenue operations for AMI utilities: tariffs, financial bills, payments, reconciliation and prepayment



Concept illustration; not a customer installation or runtime evidence.

Positioning

AetherRevenue is the revenue-operations component of AetherONE and sits after AetherMDMS in the AMI chain. In the approved design, MDMS turns validated meter data into technical Bill Determinants, and AetherRevenue is responsible for the financial work that follows. That work is applying tariff versions to the determinants, running billing, issuing financial bills, allocating payments, reconciling those payments against bank and gateway records, tracking debt, and recording bill runs and payments in the revenue subledger. For prepaid customers the same design covers the account and its balance ledger, the merchant network and its settlement, the vending transaction record, and the reference metadata of the STS token lifecycle.

The borders with neighbouring products are drawn on purpose. AetherMDMS owns usage and Bill Determinants, and a Bill Determinant is a technical billing input, never a bill. AetherCIS owns the customer account, including its identity and lifecycle, as well as the premise, Service Point and agreement. AetherRevenue owns the financial side: tariff, billing, payment, debt, settlement and prepayment. Its versioned contracts carry the account references that billing and payment use. Neither product is designed to read the other's private database.

Intended readers

- **Billing analysts and supervisors**, who need tariff versions, the exceptions held back from a bill run, and the approval trail before a run is released.
- **Payment-allocation and reconciliation operators**, who match payments to bills and clear what bank and gateway records leave unmatched.
- **Prepayment and merchant operations teams**, who follow a vending transaction from sale to meter confirmation and compare meter and account figures.
- **AMI, CIS, MDMS and finance integration teams**, who need to know which product writes which record and which versioned contract carries it.

Responsibility and capability

The table lists what the approved design makes AetherRevenue responsible for, with the objects and decisions each area involves.

Area	AetherRevenue responsibility	Objects and decisions
Tariffs	Catalogue, versions, design, simulation against golden cases, assignment periods	Tariff version, effective period, expected against calculated amount, the tariff assigned to an account
Billing accounts and cycles	Billing account list and 360 view, bill-cycle calendar	Billing account, cycle, billing window
Bill runs	Population, billed amount, quarantined exceptions, approval trail	Run, exception row and its reason, approval decision
Bills and adjustments	Bills with a step-by-step calculation trace; rebills, credit notes and other adjustments	Bill, calculation trace, adjustment with reason and status
Payments and allocation	Payments by channel and their allocation to bills	Payment, allocated and unallocated amount
Reconciliation	Matching of bank statement lines and gateway settlements	Matched item, unmatched item, hint for follow-up
Debt and service actions	Aging buckets, collection cases, approval of disconnect and reconnect requests	Arrears bucket, collection case, service-action request
Subledger and finance export	Revenue subledger journals; export batches with acknowledgement towards a separate AetherFinance boundary	Journal, export batch, acknowledgement
Prepayment accounts	Prepayment account, append-only balance ledger, meter reconciliation	Balance entry, comparison of ledger and meter register
Merchants	Merchant, outlet and terminal hierarchy, credit, commission and settlement	Merchant, terminal, settlement period
Vending	Vending console and transactions, token reissue, reversal and refund workflows	Vending transaction and its lifecycle
STS lifecycle and security context	Token lifecycle, SGC, KRN, KEN and TID metadata, a projection of security-module status, a key-change campaign view	Reference metadata only, with no token value or key
STS training lab	Fixed, publicly displayable training vectors in a TEST SECURITY DOMAIN	Training and decoder walkthroughs

Interfaces and private-writer responsibility

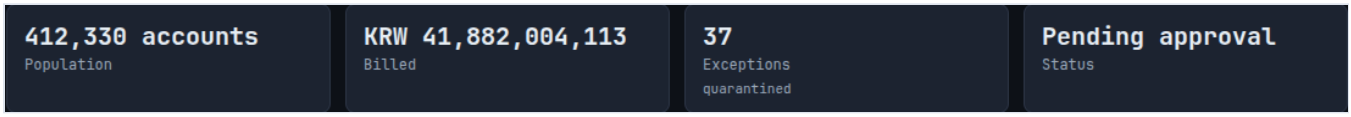
In the approved design AetherRevenue keeps its own private store and is its only writer. Other products reach it through versioned contracts and projections, and it is not designed to read the private stores of AetherMDMS, AetherHES, AetherCIS or AetherDispatch.

Contract	Direction	Carries
`mdms.bill-determinant.ready.v1`	AetherMDMS to AetherRevenue	The Bill Determinant that is ready for billing

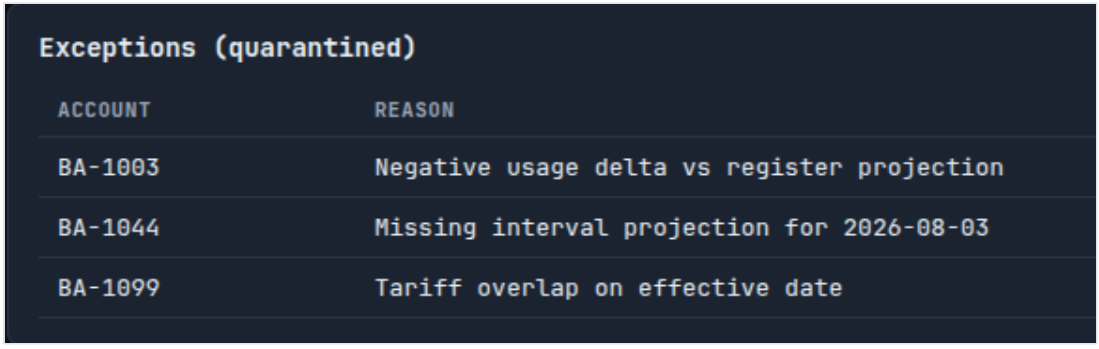
Contract	Direction	Carries
`revenue.bill.issued.v1`	From AetherRevenue	The issued bill and the Bill Determinants it used
`revenue.payment.confirmed.v1`	From AetherRevenue	The confirmed payment on a billing or prepayment account
`revenue.sts-token.state-changed.v1`	From AetherRevenue	A change in state of a token intent, with a token fingerprint rather than the token

All four are defined at schema version 1.0.0. AetherCIS is the documented source of versioned customer and service references. Financial data is exported towards AetherFinance, a separate product with its own data, and AetherRevenue is designed to show only the status of that contract. Bank and payment-gateway links, meter confirmation of a vend, and the CIS references are items for a project's integration plan.

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



2 Quarantined exceptions with their reasons



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



PRODUCT SCREEN · DEMONSTRATION DATA Evaluation interface with demonstration data. Three excerpts of one synthetic bill run. Accounts, KRW amounts and times are examples, and the run is not a real bill run.

Edition and conditions

The responsibilities above are the approved design. What this edition makes available is an evaluation interface for the same areas, so that reviewers can walk through the screens and the responsibilities behind them. It opens by default without a data source. Choosing the demonstration mode loads a fixed set of synthetic records, with KRW amounts, accounts and times chosen as examples. The interface can also show loading, empty, error, permission-denied and offline situations for review.

The views are walkthroughs, and nothing in this edition performs a business operation. There is no connected API, authentication, persistence or posting, and the approval, vending, rebill, refund, reissue and campaign controls say on screen that nothing was executed. The bills shown are financial bills in a demonstration, and no tax-invoice format or compliance is claimed. The STS pages display metadata and references only, and the training lab uses fixed public vectors. They do not generate tokens, hold keys or control a security module, and they make no claim of STS certification or production use.

A utility deployment would have to settle the tariff rules of its jurisdiction, the payment channels, the finance integration, authentication and authorization, persistence, and the contract versions with MDMS and CIS. This datasheet publishes no capacity or availability figure, no protocol compatibility statement and no customer reference.

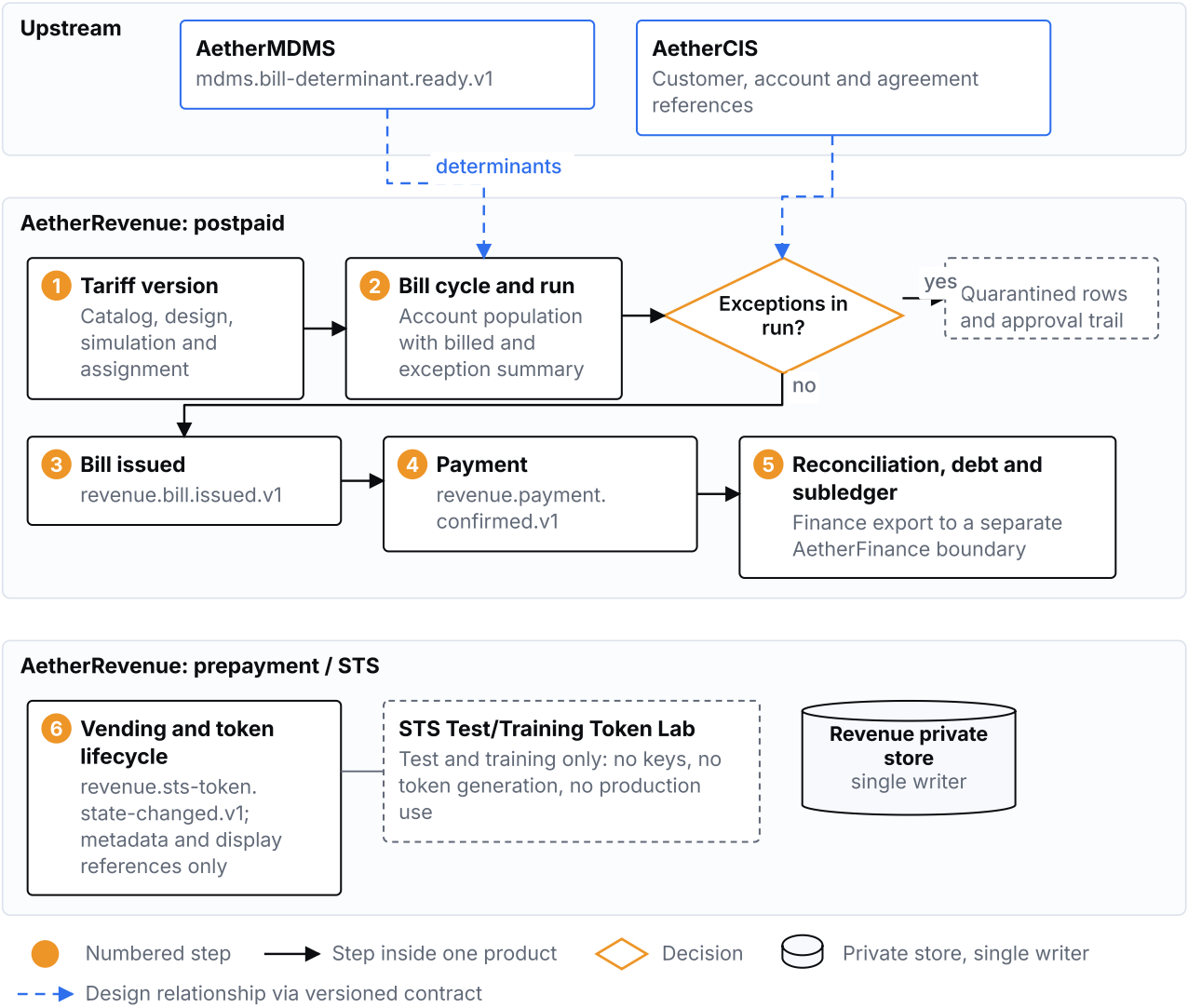
Technical specifications

- **Interface.** Web interface built with React, TypeScript and Vite, with demonstration records kept separate from the default view.
- **Contracts.** Four versioned event contracts at schema version 1.0.0, listed above.
- **Currency in demonstration records.** KRW, as synthetic examples.
- **Document edition.** 2026.10 identifies this document. It is not a commercial release.

AetherRevenue: from determinant to reconciled payment

DESIGN WORKFLOW

Design workflow; amounts, billing versions and transitions shown in the UI are demonstration data.



DESIGN WORKFLOW Design workflow. It shows the intended order of responsibility from Bill Determinant to reconciled payment, and the single private writer. It does not describe a running integration.