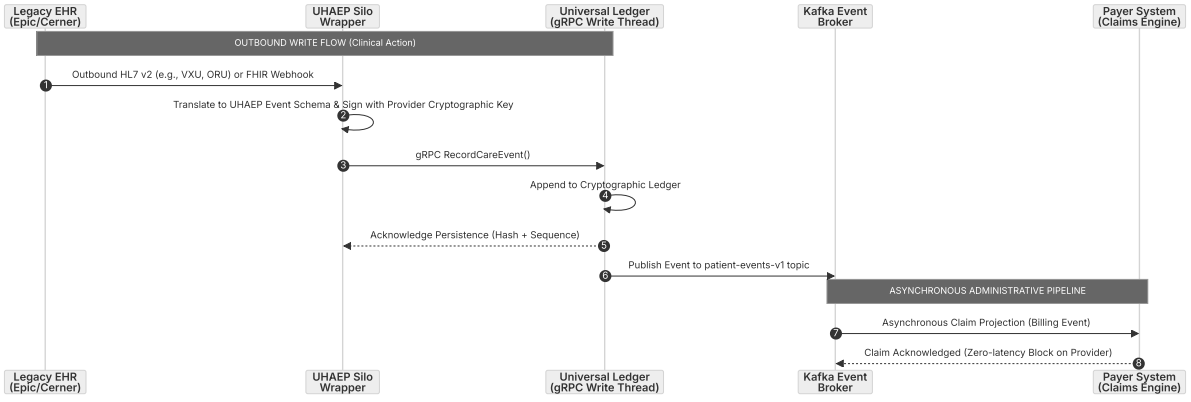
The Universal Health Architecture & Exchange Protocol (UHAEP)

Part II: Technical Specification & Integration Blueprint

Document Reference: `SB-UHAEP-2026-02-TEC`
Author: Aventine Labs LLC
Date: July 7, 2026

1. Legacy Integration: Plug-and-Play Silo Wrappers

The Universal Health Architecture & Exchange Protocol (UHAEP) does not require the immediate removal of existing enterprise hospital stacks (e.g., Epic, Cerner, Meditech). Instead, these relational platforms are treated as private internal state silos that interface with the universal ledger via a standardized API Facade and translator wrapper layer.



The Silo Wrapper executes two primary loops:

- **Outbound Loop (Clinical Write Interceptor)**: Intercepts legacy clinical transactions (HL7 v2 ADT/ORM/ORU or FHIR REST calls), serializes the clinical details, signs the payload using the provider's HSM-backed key, and appends it to the UHAEP ledger via gRPC.
- **Inbound Loop (Ledger Sync Engine)**: Subscribes to the patient's Kafka event stream. Upon receipt of a new ledger event written by an external clinic or laboratory, the wrapper translates it back into HL7/FHIR formats and injects it into the legacy EHR's ingestion interface. This updates the local database and keeps historical files in sync.

---

2. JSON Event Schemas

All clinical events are serialized as structured JSON payloads. These payloads are encrypted at the cell level within the ledger using the patient's AES-256-GCM symmetric key.

Schema 1: `MedicationAdministration`

```
{
  "$schema": "https://json-schema.org/draft/2020-12/schema",
  "title": "MedicationAdministrationEvent",
  "type": "object",
  "required": [
    "eventId",
    "timestamp",
    "patientId",
    "medication",
    "dosage",
    "route",
    "practitionerNpi"
  ],
  "properties": {
    "eventId": { "type": "string", "format": "uuid" },
    "timestamp": { "type": "string", "format": "date-time" },
    "patientId": { "type": "string", "description": "Patient Sovereign Root DID" },
    "medication": {
      "type": "object",
      "required": ["rxNormCode", "display"],
      "properties": {
        "rxNormCode": { "type": "string", "pattern": "^[0-9]+$" },
        "display": { "type": "string" }
      }
    },
    "dosage": {
      "type": "object",
      "required": ["value", "unit"],
      "properties": {
        "value": { "type": "number" },
        "unit": { "type": "string" }
```

```
      }
    },
    "route": { "type": "string", "enum": ["IV", "PO", "IM", "SC", "PR", "TOP"] },
    "practitionerNpi": { "type": "string", "pattern": "^[0-9]{10}$" }
  }
}
```

Sample Data: `MedicationAdministration`

```
{
  "eventId": "9b1deb4d-3b7d-4bad-9bdd-2b0d7b3dcb6d",
  "timestamp": "2026-07-07T18:05:00Z",
  "patientId": "did:uhaep:patient:123456789",
  "medication": {
    "rxNormCode": "866107",
    "display": "Lisinopril 10 MG Oral Tablet"
  },
  "dosage": {
    "value": 1.0,
    "unit": "Tablet"
  },
  "route": "PO",
  "practitionerNpi": "9876543210"
}
```

Schema 2: `DiagnosisAsserted`

```
{
  "$schema": "https://json-schema.org/draft/2020-12/schema",
  "title": "DiagnosisAssertedEvent",
  "type": "object",
  "required": [
    "eventId",
    "timestamp",
    "patientId",
    "diagnosis",
    "clinicalStatus",
    "practitionerNpi"
  ],
  "properties": {
    "eventId": { "type": "string", "format": "uuid" },
    "timestamp": { "type": "string", "format": "date-time" },
    "patientId": { "type": "string" },
    "diagnosis": {
      "type": "object",
      "required": ["icd10Code", "display"],
      "properties": {
        "icd10Code": { "type": "string", "pattern": "^[A-Z][0-9][0-9A-Z](\\.[0-9A-Z]-
        "display": { "type": "string" }
      }
    },
    "clinicalStatus": { "type": "string", "enum": ["active", "recurrence", "remissior
    "practitionerNpi": { "type": "string", "pattern": "^[0-9]{10}$" }
  }
}
```

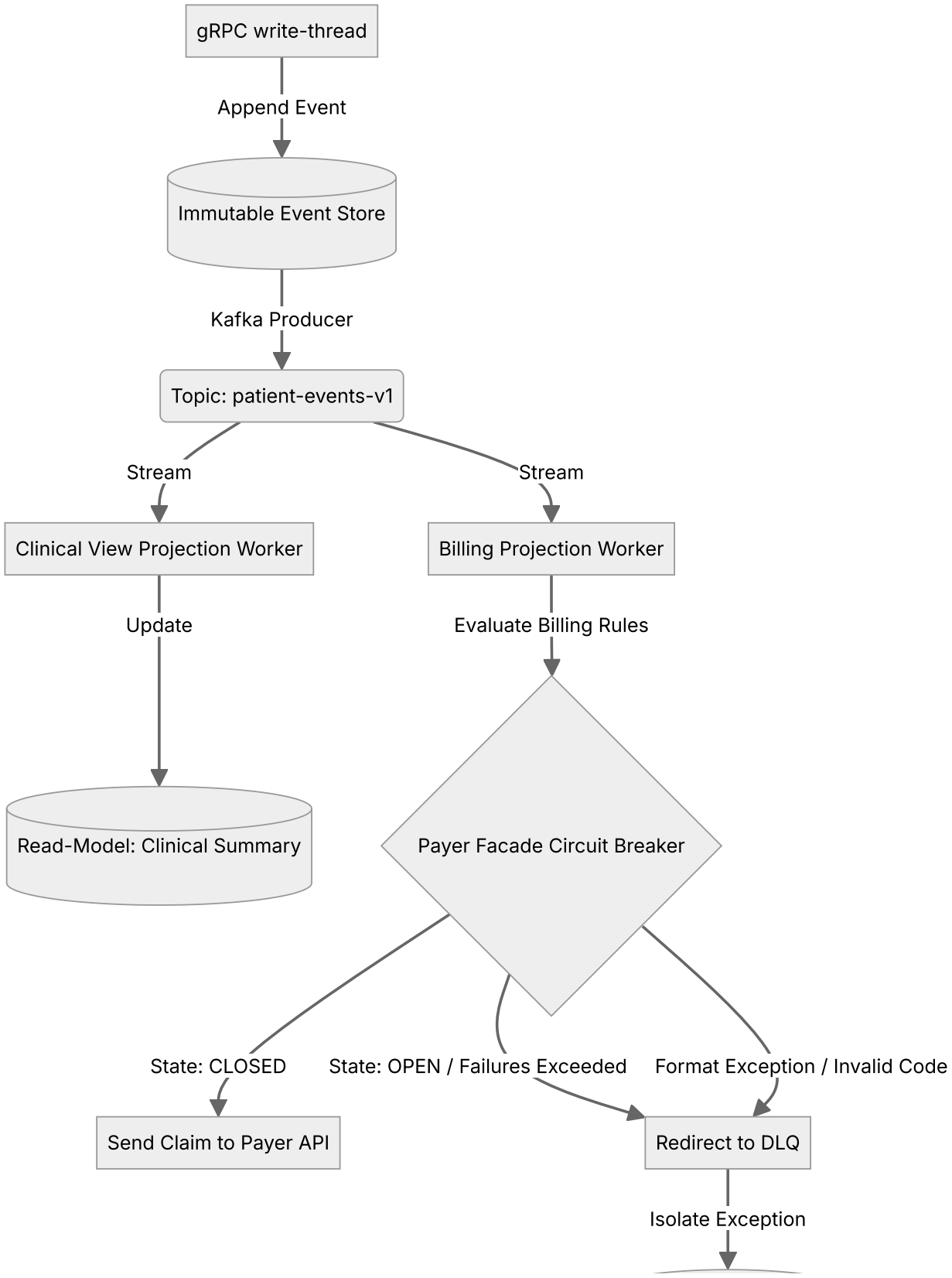
Sample Data: `DiagnosisAsserted`

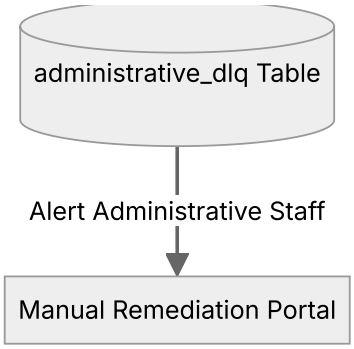
```
{
  "eventId": "0f8e9d0c-1b2a-3c4d-5e6f-7a8b9c0d1e2f",
  "timestamp": "2026-07-07T18:06:00Z",
  "patientId": "did:uhaep:patient:123456789",
  "diagnosis": {
    "icd10Code": "I10",
    "display": "Essential (primary) hypertension"
  },
  "clinicalStatus": "active",
  "practitionerNpi": "9876543210"
}
```

---

3. Asynchronous CQRS & DLQ Boundary Rules

The UHAEP pipeline enforces strict isolation of the clinical write loop from administrative systems (e.g., prior authorization, billing, and regulatory audits) by utilizing Apache Kafka as the backbone event broker.


gRPC write-thread
Append Event
Immutable Event Store
Kafka Producer
Topic: patient-events-v1
Stream
Stream
Clinical View Projection Worker
Billing Projection Worker
Update
Evaluate Billing Rules
Read-Model: Clinical Summary
Payer Facade Circuit Breaker
State: CLOSED
State: OPEN / Failures Exceeded
Format Exception / Invalid Code
Send Claim to Payer API
Redirect to DLQ
Isolate Exception



Dead Letter Queue (DLQ) Boundary Rules:

1. **Clinical Non-Block**: An exception written to the `administrative_dlq` must never trigger an exception in the clinical write-thread. The clinical system must receive a HTTP 201/gRPC OK success response immediately upon ledger persistence, regardless of downstream billing failures.
2. **Micro-Isolation**: Billing and prior authorization validation rules must operate on individual event granularities. A failed prior authorization check on a single clinical event (e.g., an MRI scan) will divert that specific event's billing projection to the DLQ, but is blocked from holding up processing or payment of other undisputed items (e.g., physician consult fees) in the same care session.
3. **Payer Interface Circuit Breaker Configuration**:
    - **Failure Rate Threshold**: 5% of outgoing payer API queries over a rolling window of 60 seconds triggers the circuit breaker to OPEN.
    - **Open State Behavior**: Incoming billing projections immediately bypass the payer API and route directly to the billing queue with status `PAYER_UNAVAILABLE_PENDING`.
    - **Automatic Retry**: After 30 seconds in the OPEN state, the breaker enters HALF-OPEN, routing a single test transaction. If successful, the breaker resets to CLOSED.