

WHITEPAPER · TECHNICAL SPECIFICATION

Indoor & Outdoor Weather Data Ecosystem on On-Chain Exchange Infrastructure

A nationwide IoT weather sensing network - two layers, **indoor (IAQ)** and **outdoor (OAQ)** - collects living environment data, supplies it to appliance manufacturers for R&D and marketing and to environmental services, and returns on-chain rewards to contributors. **Part 1** is the business whitepaper; **Part 2 onward** is the XRPL DePIN · x402 technical specification that implements it.

IAQ PM · CO₂ · TVOC · temp/humidityOAQ PM · O₃ · NO₂ · SO₂ · CO

XLS-20 device licensing

IOU Wellbian (WLBN) reward token

XLS-30 AMM liquidity

HTTP 402 data API billing

WP 1

Introduction

People spend more than 90% of the day indoors, and indoor weather is directly tied to personal health. Yet most indoor climate appliances - air purifiers, dehumidifiers, air conditioners - operate on standard algorithms fixed at the factory.

This project builds an **indoor climate data platform ecosystem**: a nationwide IoT indoor weather sensing network that collects users' actual living environment data in real time and supplies it to appliance manufacturers, supporting both engineering advancement and hyper-personalized marketing.

90%+

of the day the average person spends indoors

0

post-sale living environment data manufacturers currently hold

6+7

metrics measured continuously - 6 indoor (IAQ) · 7 outdoor (OAQ)

HOW THIS DOCUMENT IS ORGANIZED

Part 1 (Whitepaper) describes the business structure of the ecosystem. **Parts 2–3 (Technical Overview · Weather Data Spec)** and **Steps 1–6 and the Operations sections** cover the data schemas, licensing and reward design, de-identification pipeline, API billing protocol, and step-by-step implementation guidance needed to build that structure on XRPL. Each whitepaper chapter links to its corresponding technical section.

WP 2

Problem Statement and Market Limitations

PROBLEM 01

No real-world data at the manufacturer

After a product ships, manufacturers have little visibility into how indoor weather actually changes in customers' homes, how usage patterns shift by season, or how climate-control efficiency varies with the living environment - floor area, location, daylight exposure.

PROBLEM 02

Uniform product development and marketing

Required dehumidification capacity and air purification performance differ by region and housing type, yet the market still competes on one-size-fits-all specs and broad, untargeted marketing.

WHAT THE DATA GAP COSTS

Without real-environment data, manufacturers either **over-engineer to worst-case conditions** or ship products that underperform in specific housing types. On the marketing side, there is no way to identify "households that need dehumidification," so

campaigns fall back on broadcasting to the entire customer base - which drives up cost per lead. This ecosystem fills those two gaps with [R&D data](#) and a [cohort API](#) respectively.

WP 3

Platform Ecosystem Structure

The ecosystem comprises three parties, with data collection, refinement, and utilization forming a virtuous cycle.

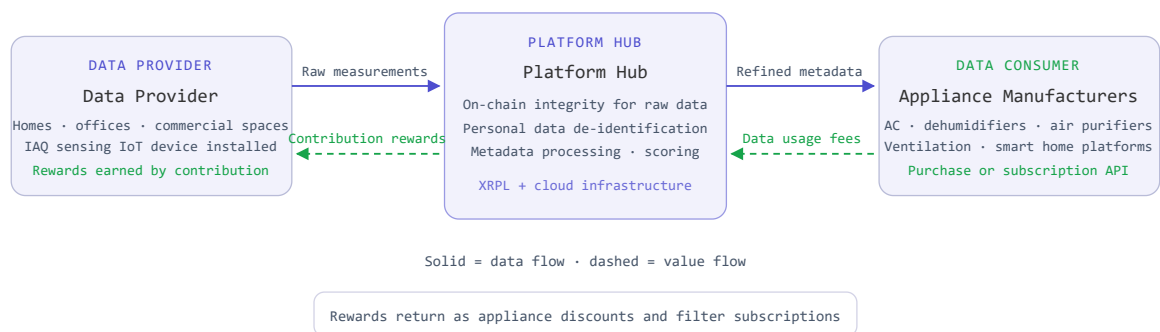


Fig 0. Three-party virtuous cycle

3.1 Data Provider

- Users who install indoor (IAQ) monitors in homes, offices, and commercial spaces, and outdoor (OAQ) monitors on rooftops, roadsides, and school grounds.
- They supply real-time environmental data to the platform and earn rewards proportional to their contribution.

3.2 Data Platform Hub

- Stores collected raw data tamper-proof through blockchain and cloud infrastructure.
- De-identifies (anonymizes) personal data, then processes it into metadata that manufacturers can act on.

3.3 Appliance Manufacturers

- Companies that manufacture air conditioners, dehumidifiers, air purifiers, ventilation systems, and similar products.
- They purchase refined data or receive it via subscription APIs, and apply it to R&D and marketing.

TECHNICAL MAPPING

Contributor credentials → [XLS-20 license NFT](#) · raw data integrity → [oracle batch Merkle root](#) · de-identification → [5-stage pipeline](#) · consumer billing → [x402 data products](#).

WP 4

Collected Data and Core Applications

Collected data is classified and delivered according to the buyer's purpose: **engineering data (R&D)** and **marketing data (CRM/Sales)**.

4.1 Indoor (IAQ) metrics - appliance R&D · marketing

METRIC	RELATED APPLIANCES	ENGINEERING USE (R&D)	MARKETING USE (SALES)
Temperature & humidity (RH · derived absolute humidity)	Dehumidifiers, air conditioners	Develop optimal dehumidification algorithms per housing type; build condensation-prevention prediction models	Target monsoon-season high-humidity households for dehumidifier promotions; alert dry households on humid replacement timing
Particulate matter (PM10 / PM2.5)	Air purifiers, vacuum cleaners	Refine filter life prediction models; design automatic fan-speed control when pollution spikes	Identify households with indoor pollution track outdoor levels; target for high-capacity purifiers
Carbon dioxide (CO ₂)	Ventilating purifiers (ERV)	Develop AI ventilation-timing prediction from CO2 rate-of-change in enclosed spaces	Propose smart ventilation systems to under-ventilated public facilities and households with infants

Metric	Related Appliances	Engineering Use (R&D)	Marketing Use (Sales)
TVOC (total volatile organic compounds)	Sick-building-syndrome removers	Analyze off-gassing patterns by building material; optimize sensor sensitivity	Market specialty filter new-apartment move and households undergoing interior w
Temperature / apparent temperature	Air conditioners, heaters	Apparent-temperature-based setpoint control logic; seasonal heating/cooling load profiling	Replacement propose heating/cooling appliances, targeted a households with sust extreme-heat or extre cold comfort conditio

4.2 Outdoor (OAQ) metrics - environmental services · outdoor-linked control

Outdoor metrics have a different buyer mix. Where indoor metrics center on appliance manufacturers, outdoor metrics have **municipalities and environmental services as primary buyers**, with manufacturers as secondary buyers in the form of "control that reacts to outdoor conditions".

Metric	Primary Users	Engineering Use (R&D)	Service Applications
Particulate matter (PM10 / PM2.5)	Appliance manufacturers · municipalities	Derive per-cell infiltration (I/O ratio); develop preemptive intake/shutoff control tied to outdoor conditions	Alerts to close windows and pre-run air purifiers based on outdoor deterioration forecasts
Ozone (O ₃)	Municipalities · public health	Analyze photochemical formation patterns; build a model for when ventilation must be suppressed	Advise against ventilation during ozone advisories - send control signals to ventilating purifiers
Nitrogen dioxide (NO ₂)	Municipalities · transport	Isolate roadside emission contribution; map roadside exposure	Low-exposure route guidance for school and pedestrian paths
Sulfur dioxide (SO ₂)	Industrial complexes · environment	Estimate the impact radius of industrial sources;	Emission-event alerts to nearby residential areas

Metric	Primary Users	Engineering Use (R&D)	Service Applications
		validate dispersion models by wind direction	
Carbon monoxide (CO)	Municipalities · safety	Analyze accumulation patterns in traffic congestion zones	Short-term exposure warnings in dense areas
Wind speed · direction	All parties	Estimate pollutant transport and dispersion direction; trace back to emission sources	Predict pollutant inflow from adjacent cells

PM2.5 IS WHAT LINKS THE TWO TABLES

PM2.5 is the only metric appearing in both 4.1 and 4.2, because the same substance is measured indoors and outdoors by the same light-scattering method. Every other metric exists on only one side and is a standalone product, but PM2.5 multiplies the two networks into a **derived metric: the infiltration ratio**. That derived metric is why [Phase 3](#) holds the most valuable data - and it would not exist had the outdoor layer been a weather station.

PRECONDITIONS FOR MARKETING USE

Every marketing item in the table above takes the form of "**extract households** meeting a given condition." This must be implemented as an API that **returns cohort counts only**, never identifying individuals. Actual ad delivery is brokered by the platform; the manufacturer never receives a subject list. The concrete design is covered in the [de-identification pipeline](#) and [cohort API](#) sections.

WP 5

Data Incentive Mechanism

Securing broad, meaningful data on an ongoing basis depends on users voluntarily operating their IoT devices.

EARN

Contribution rewards

Nodes (monitors) that stream high-quality data around the clock receive a Wellbian (WLBN) reward every day (epoch = 1 day, KST midnight) as a fixed quantity. The daily reward is defined purely in quantity - $\text{base} \cdot 7 \times 0.78^{(\text{phase}-1)} \times \text{quality grade} \cdot (0-1.2)$ - never in value.

SPEND

Ecosystem spending

Earned rewards are spendable on appliance purchase discounts, filter replacement subscriptions, or the platform's partner marketplace - by design, to prevent churn.

WHY "HIGH-QUALITY DATA" HAS TO BE QUANTIFIED

If rewards scaled only with submission count, the optimal strategy would be to neglect or spoof the sensor to hit the count. This specification assigns a quality grade (0 · 1.0 · 1.2) from a transmission-rate gate plus a [Quality Score](#) combining uptime, outlier rate, neighbor-node correlation and calibration age, then **fixes the epoch budget, pays fixed quantities by grade, and redistributes or burns the unpaid share**. Total emission is fixed in advance by [phase gates and the ten-phase cap \(2.665B\)](#).

LEGAL NOTICE

WLBN and the license NFT confer no equity, dividend, profit-sharing, or repayment claim against the company. Rewards are a utility token distributed by the quality of verified data contributions; the amount, value, and liquidity of any payout are not guaranteed. Every figure in this document is a design and verification assumption and is in no case a prediction or promise of returns. Purchasing a device is the purchase of data-collection equipment, not an investment contract.

WP 6

Business Revenue Model

REVENUE 01

B2B data API subscriptions

Regional indoor weather statistics APIs offered to appliance manufacturers and smart home platforms (SmartThings, ThinQ, etc.) on monthly or annual subscription.

REVENUE 02

Targeted advertising fees

Per-lead fees charged when a manufacturer uses platform data to advertise tailored products to users meeting specific environmental criteria.

REVENUE 03

Insight report sales

Analytical reports on indoor climate-control efficiency by floor plan and material, sold to construction and interior design firms.

All three revenue streams map onto [x402 data products](#) - subscriptions become prepaid credits, lead fees are billed per cohort-API call, and reports are billed per LLM inference. Since billing is unified under a single protocol, settlement works without a separate billing system. On top of this, [Data Bounty](#) adds a reverse channel where buyers deposit funds first and request data.

WP 6+

Data Bounty - Data That Extends Lives

Modern people spend **about 90% of their day indoors**, and the WHO estimates premature deaths from indoor air pollution at several million per year. Measuring indoor weather is not a convenience feature - it is **where prevention of respiratory and cardiovascular disease begins**. The minute-by-minute data node operators accumulate every day is public-health infrastructure that turns living spaces that "went unmeasured and thus without intervention" into manageable ones, and this ecosystem's rewards are not a mere mining scheme but compensation for that contribution. Data Bounty is the mechanism that directly connects those who need this contribution with data providers.

BOUNTY 01

Prepaid Escrow

Companies and institutions that need data (appliance makers, HVAC and ventilation solution providers, research institutes, local governments) **deposit XRP or RLUSD up front** to open a bounty. Only bounties with a confirmed on-chain deposit are shown to nodes, which structurally reduces the risk of non-payment.

BOUNTY 02**Exclusive Data Usage Rights**

Unlike the public API anyone can purchase, **a dedicated access key is issued once, only to sponsors who complete the deposit and join the bounty**, and only this key grants access to participating nodes' data for the bounty period. The query scope is limited to consenting participant nodes × the bounty period × each node's data from its join time onward, and participating nodes receive bounty rewards on top of their base epoch rewards.

BOUNTY 03**Clean-Air Solution Promotions**

Nodes that **opt in to marketing communications** when joining may receive promotions for the sponsor's indoor-weather (weather) improvement solutions (air purifiers, ventilation systems, filter subscriptions, and so on). Consent is per bounty and can be withdrawn at any time.

THE REVENUE LOOP THE BOUNTY COMPLETES

If the three revenue streams of WP 6 are a forward flow - selling products the platform builds - Data Bounty is a **reverse flow in which buyers stake funds and request data**, and what they receive is not a mere API call entitlement but **time-limited data usage rights that accrue only to subscribed sponsors**. Sponsor deposits are transparently verifiable on XRPL mainnet, and are settled by participation share when the bounty period ends. Those who measure earn more, those who need data get verified field measurements, and clean-air solutions reach exactly the households that need them - a closed loop where data leads to health interventions. The bounty board runs on the execution platform wellbian.io/data.

WP 7

Development Roadmap

PHASE 1**Build initial IoT infrastructure and secure an early-adopter data collection network - complete (XRPL mainnet live, Aug 2026)**

Device license NFT issuance, telemetry collection pipeline, and quality-score-based reward engine are running on XRPL mainnet. Corresponds to [Steps 1–4](#) of the technical specification.

PHASE 2**Sign MOUs with major appliance manufacturers and move data API integration to production (in progress)**

Validate the de-identification pipeline, open the cohort and aggregate APIs, and integrate the x402 billing gateway (RLUSD). Corresponds to [Steps 5–6](#) and the [B2B product design](#).

PHASE 3**Combine with the outdoor OAQ network for an indoor–outdoor linked smart home control solution**

Capturing outdoor and indoor weather in the same cell enables predictive, preemptive climate control. Both networks share identical licensing, reward, and billing infrastructure, distinguished only by `NFTokenTaxon`.

WHAT PHASE 3 MEANS TECHNICALLY

Outdoor OAQ is the **leading indicator**; indoor IAQ is the **lagging one**. PM2.5 in particular is **the same substance measured by both networks**, so capturing both time series in the same cell allows the infiltration (I/O ratio) transfer function to be learned directly - "when outdoor PM is X, indoor PM in this housing type moves to Y". This is unobtainable when the outdoor layer is a weather station, and it is the most valuable form of data a manufacturer can buy. It is why this specification puts both networks on one ledger and one billing system.

Technical

Platform Overview

Implementing the whitepaper's three-party structure on-chain connects two economic loops on a single ledger. The **supply side** is a DePIN loop where IAQ nodes upload indoor environment data and receive token rewards. The **demand side** is an x402 loop where appliance manufacturers and AI agents call data APIs and pay tokens. The two loops are linked in liquidity through an AMM pool on the same ledger.

SUPPLY LOOP · DEPIN**Upload data, receive tokens**

When an authorized IoT device (holding a license NFT) transmits IAQ measurements, a quality score is computed and accrued per 1-day (KST) epoch; after a 24-hour hold the Hot Wallet pays out WLBN. More nodes = denser sensing network.

DEMAND LOOP · X402

Use data, pay tokens

When manufacturers or AI agents call the aggregate, cohort, or report APIs, the server demands payment via HTTP 402, verifies the on-chain transaction hash, and returns the response. Call volume = token demand.

Two networks, one infrastructure

The indoor IAQ network and the outdoor OAQ network differ only in sensor configuration and data schema; licensing, reward payout, and data billing paths are identical. In implementation they are distinguished solely by `NFTokenTaxon` and telemetry schema version - wallets, tokens, AMM, and the x402 gateway share a single instance.

TYPE	INDOOR IAQ NETWORK	OUTDOOR OAQ NETWORK
Metrics	PM2.5/10, CO ₂ (NDIR), TVOC, temp/humidity (feels-like derived)	PM2.5/10, O ₃ , NO ₂ , SO ₂ , CO, temp/RH, wind speed/direction
Installed by	Households · offices · commercial spaces	Building rooftops · roadsides · schools · near industrial complexes
<code>NFTokenTaxon</code>	<code>3026</code>	<code>2026</code>
Primary buyers	Appliance manufacturers · smart home platforms	Municipalities · environmental services · AI agents
Shared infrastructure	Issuer/Hot wallets · WLBN·RLUSD · AMM pool · x402 gateway · burn batch	Same (single shared instance)

Five propositions verified on mainnet

#	PROPOSITION	VERIFICATION METHOD	XRPL FEATURE
1	Device identity can be proven on-chain and cannot be forged	<code>verifyDeviceLicense()</code> on-chain lookup	XLS-20 NFT

#	PROPOSITION	VERIFICATION METHOD	XRPL FEATURE
2	Micro-rewards for data contribution can be fully automated	Epoch settlement → batch Payment (24h claim hold)	Issued Asset (IOU)
3	Swaps between the token and a stablecoin can happen within the ledger	Live RLUSD ↔ WLBN swap in the mainnet AMM pool	XLS-30 AMM
4	An AI agent can pay API usage fees with no human involvement	402 response → payment → retry → 200	Payment + tx lookup
5	Usable cohorts can be extracted for manufacturers without identifying individuals	Reject the response when k-anonymity is unmet; return counts only	Off-chain (not recorded on-chain)

WHY XRPL

For micropayments to work, **fees must be overwhelmingly smaller than the payment itself**. XRPL's base transaction cost is 10 drops (0.00001 XRP) with finality in 3–5 seconds. That is why 0.002 RLUSD aggregate-API charges and epoch batch reward payouts across tens of thousands of nodes survive without being eaten by fees. NFTs, IOUs, and AMM are also provided as **native transaction types** rather than smart contracts, keeping the auditable code surface small, which lowers operational risk.

Technical

System Architecture

Three off-chain layers (device · gateway · consumer) plus one on-chain layer (the XRPL ledger).

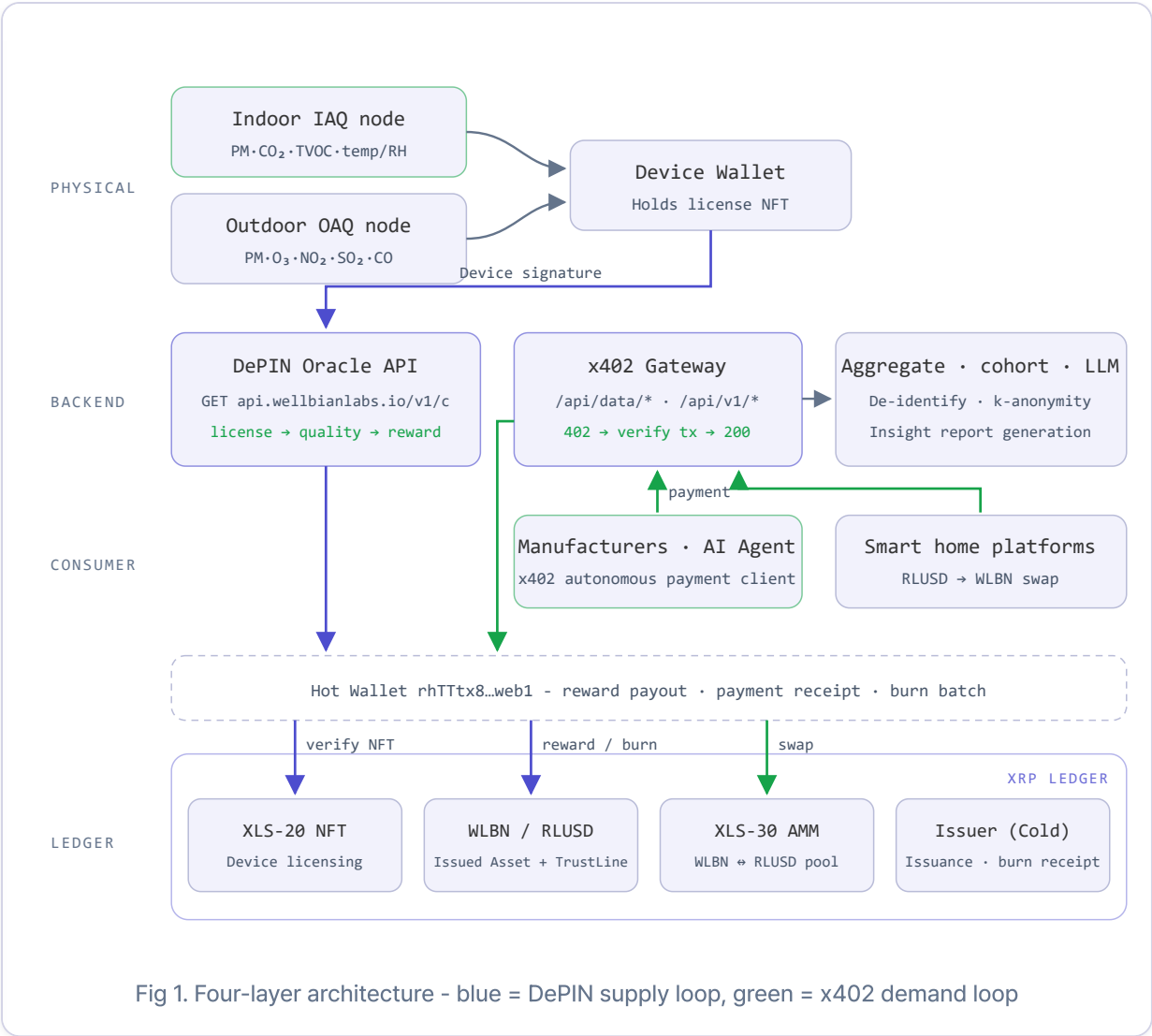


Fig 1. Four-layer architecture - blue = DePIN supply loop, green = x402 demand loop

Wallet role separation

WALLET	ROLE	KEY
Issuer (Cold) rDJz8WJhsKgydqJzSZXMpsJot3eRmSkR5	Issuer of Wellbian (WLBN), NFT minter, burn recipient. RLUSD is issued by Ripple (rMxCKbEDwqr76QuheSUMdEGf4B9xJ8m5De) and accepted via TrustLine.	Off ass
Hot Wallet rhTTtx8YyKzcmPDSoroYXuHdGMThrnweb1	Epoch reward payout, x402 payment receipt, burn batch execution	Se env var rec
IoT Device (ARC-600DA)	Device wallet, license NFT holder	De ele ass

WALLET	ROLE	KEY
AI Agent · manufacturer client	x402 client, API fee payer	Ag

COLD / HOT SEPARATION PRINCIPLE

A leaked Issuer key allows unlimited issuance, so the runtime server **never loads the Issuer seed**. Issuance happens only in a one-off bootstrap script; every ongoing transaction thereafter is handled by the Hot Wallet. This boundary is enforced at code level - separate modules plus separately injected keys - and the production server never loads the Issuer seed.

Technical

Stack & Network

LANGUAGE

TypeScript / Node.js 20+

Runs on the Vercel Runtime. This document is written against TypeScript.

XRPL SDK

xrpl.js v3+

Uses the WebSocket `Client`, `Wallet`, and autofill/sign/submitAndWait.

API

Next.js 15 App Router (Vercel)

The x402 gateway is implemented as Route Handler middleware. Batch and epoch settlement run on Vercel Cron.

Network parameters

ITEM	VALUE	NOTES
WebSocket	wss://xrplcluster.com	Public mainnet cluster node

ITEM	VALUE	NOTES
JSON-RPC	https://xrplcluster.com	For one-off queries
Account activation	Payment (external deposit)	Account activation · pre-fund XRP above the Base Reserve (no faucet)
Explorer	livenet.xrpl.org	Visual verification of tx / account
Base Reserve	1 XRP	Minimum balance to keep an account alive (mainnet)
Owner Reserve	0.2 XRP / object	Charged per TrustLine · NFT Offer · AMM object
Finality	3–5 seconds	Async waiting and polling are mandatory in the design

Currency code encoding

XRPL's currency field must be either 3-character ASCII or **40-character hex (160-bit)**.

Both [WLBN](#) and [RLUSD](#) exceed three characters, so hex encoding is required. Without pinning these as constants, TrustSet and Payment currencies diverge and you get

[tecNO_LINE](#) / [tecPATH_DRY](#) .

src/config/currency.ts

TYPESCRIPT

```
// ASCII → 20-byte hex (right-padded with zeros)
export const toCurrencyHex = (code: string): string => {
  if (code.length <= 3) return code; // 3 chars or fewer pass through
  return Buffer.from(code, 'ascii').toString('hex').toUpperCase().padEnd(40, '0');
};

export const WLBN = toCurrencyHex('WLBN'); // 574C424E00000000000000000000000000000000
export const RLUSD = toCurrencyHex('RLUSD'); // 524C555344000000000000000000000000000000
```

Environment variables

.env.example

DOTENV

```

XRPL_ENDPOINT=wss://xrplcluster.com
XRPL_NETWORK=mainnet

# Cold - bootstrap script only. Never inject into the runtime server
ISSUER_SEED=sEd.....

# Hot - API server runtime
HOT_SEED=sEd.....
HOT_WALLET_ADDRESS=r.....

# Counterparties
TREASURY_ADDRESS=rPZFY2yLgqjmBrxtxWHXnB1doBt7gu7xin
RLUSD_ISSUER_ADDRESS=rMxCKbEDwqr76QuheSUMdEGf4B9xJ8m5De

# Pricing policy (RLUSD)
X402_PRICE_AGGREGATE_RLUSD=0.002
X402_PRICE_COHORT_RLUSD=0.05
X402_PRICE_REPORT_RLUSD=0.5
X402_DEST_TAG=402001

# Reward policy
REWARD_BASE_DAILY_WLBN=7
PHASE_DECAY=0.78
EXCELLENT_BONUS_WLBN=1.4
NODE_DAILY_CAP_WLBN=10.5
BUDGET_PER_NODE_WLBN=7.14
CLAIM_HOLD_HOURS=24
BURN_RATIO=0.5

```

HANDLING SEEDS

`.env` must be listed in `.gitignore`. Committing a production (Mainnet) seed leaks signing authority the moment the repository is exposed. In production, inject from KMS/Vault instead of `.env` and restrict signing to the Hot Wallet key.

Technical

Module Structure

Each XRPL capability is isolated into its own module so it can be tested and swapped step by step.

```

src/
├─ config/
│   ├─ env.ts           # env loading + validation (zod)
│   └─ currency.ts      # WLBN / RLUSD hex constants
├─ xrpl/
│   ├─ client.ts        # Client singleton · reconnect · graceful shutdown
│   ├─ wallet.ts        # Step 1 - wallet create / load / fund
│   ├─ token.ts         # Step 2 - AccountSet · TrustSet · IOU Payment
│   ├─ nft.ts           # Step 3 - Mint / Offer / Accept / verifyDeviceLicense
│   ├─ amm.ts           # Step 5 - AMMCreate · AMMDeposit · path swap
│   └─ submit.ts        # shared autofill→sign→submitAndWait + tes check
├─ services/
│   ├─ oracle.ts        # Step 4 - telemetry intake · validation · reward
│   ├─ x402.ts          # Step 6 - 402 challenge · tx verification
│   ├─ llm.ts           # IAQ insight report generation (0.5 RLUSD)
│   └─ burn.ts          # Step 6 - 50% burn batch job
├─ api/
│   ├─ route.ingest.ts   # GET /v1/c?<base64> (Supabase Edge Function proxy)
│   ├─ route.data.ts     # GET /api/data/aggregates (x402)
│   └─ middleware.x402.ts # 402 gate middleware
├─ scripts/
│   ├─ 01-bootstrap-wallets.ts
│   ├─ 02-issue-tokens.ts
│   ├─ 03-mint-license.ts
│   ├─ 05-create-amm.ts
│   └─ x402-client.ts    # reference x402 client
└─ server.ts

```

Shared submit helper

Verifies in one place that every transaction returned `tesSUCCESS` and that `validated: true`. Without enforcing this wrapper, a "submitted but not yet in the ledger" state gets mistaken for success.

```
src/xrpl/submit.ts
```

```
TYPESCRIPT
```

```

import { Client, Wallet, SubmittableTransaction, TxResponse } from 'xrpl';

export class TxError extends Error {
  constructor(public code: string, public hash?: string) {

```

```

    super(`XRPL tx failed: ${code}${hash ? ` (${hash})` : ''}`);
  }
}

/** autofill → sign → submitAndWait → verify tesSUCCESS + validated */
export async function submit(
  client: Client, wallet: Wallet, tx: SubmittableTransaction,
): Promise<TxResponse> {
  const prepared = await client.autofill(tx);
  const signed = wallet.sign(prepared);
  const res = await client.submitAndWait(signed.tx_blob);

  const meta = res.result.meta;
  const code = typeof meta === 'object' && meta !== null
    ? (meta as { TransactionResult: string }).TransactionResult
    : 'UNKNOWN';

  if (code !== 'tesSUCCESS') throw new TxError(code, res.result.hash);
  if (res.result.validated !== true) throw new TxError('NOT_VALIDATED', res.result.hash);
  return res;
}

```

AQ Spec

Sensors & Telemetry Schema

Pins the metrics from Whitepaper ch. 4 to physical quantities a real node can measure and to a concrete payload format. The indoor (IAQ) and outdoor (OAQ) schemas differ only in their metric lists and share an **identical envelope structure**, each carrying its own version field.

Sensor specifications

METRIC	FIELD	SENSING METHOD	RANGE	TARGET ACCURACY
Particulate matter	pm25	Light scattering (PMS)	0 ~	±10% (≥100 range)
	pm10		1000 μg/m³	

METRIC	FIELD	SENSING METHOD	RANGE	TARGET ACCURACY
Carbon dioxide	co2	NDIR	400 ~ 5000 ppm	±(50 ppm + 3%)
Total volatile organic compounds	tvoc	MOX semiconductor	0 ~ 60000 ppb	Relative index (absolute value not guaranteed)
Temperature	tempC	Digital temp/RH combo	-10 ~ 60 °C	±0.3 °C
Relative humidity	rh	Capacitive	0 ~ 100 %RH	±2 %RH
Feels-like derived	feelsLikeC	Computed from temperature + relative humidity	-10 ~ 60 °C	Input error propagation
Absolute humidity derived	ah	Computed from temperature + relative humidity	0 ~ 60 g/m³	Input error propagation

Outdoor OAQ sensor specifications

Outdoor nodes measure regulated ambient pollutants. PM2.5 and PM10 are measured with **the same method on the same substance** as indoor nodes, so indoor and outdoor values in the same cell can be compared directly to derive the infiltration (I/O) ratio.

METRIC	FIELD	SENSING METHOD	RANGE	TIER
Particulate matter	pm25 pm10	Light scattering (PMS)	0 ~ 1000 µg/m³	oaq- standard
Ozone	o3	Electrochemical	0 ~ 500 ppb	oaq- standard
Nitrogen dioxide	no2	Electrochemical	0 ~ 500 ppb	oaq- standard

METRIC	FIELD	SENSING METHOD	RANGE	TIER
Temperature · relative humidity	<code>tempC</code> <code>rh</code>	Digital combo	-30 ~ 60 °C / 0 ~ 100 %RH	<code>oaq-</code> <code>standard</code>
Sulfur dioxide	<code>so2</code>	Electrochemical	0 ~ 200 ppb	<code>oaq-pro</code>
Carbon monoxide	<code>co</code>	Electrochemical	0 ~ 50 ppm	<code>oaq-pro</code>
Wind speed · direction	<code>wind</code> <code>windDir</code>	Ultrasonic	0 ~ 60 m/s / 0 ~ 360°	<code>oaq-pro</code>

OUTDOOR SENSORS CARRY A MUCH HEAVIER CALIBRATION BURDEN

Electrochemical gas sensors ($O_3 \cdot NO_2 \cdot SO_2 \cdot CO$) cross-react strongly with temperature and humidity and last only one to two years. Because they **drift faster** than indoor nodes, the calibration (15 pts) and correlation (20 pts) terms in the quality score are weighted more heavily outdoors, and correlation against the nearby national monitoring network (AirKorea) feeds the correction factor.

The indoor Phase-1 base reward is 7 per day (normal grade); the outdoor line is offered first to existing holders, and its reward quantity will be published when the outdoor model launches.

Why absolute humidity has to be derived

Whitepaper ch. 4 naming **absolute humidity** as the first metric is the right call. Relative humidity is temperature-dependent and cannot be used directly to compute dehumidification load. The same 60 %RH is about 10.3 g/m³ at 20 °C but about 18.2 g/m³ at 30 °C - an **1.8× difference in actual moisture**. The dehumidifier has to extract far more water in the latter case. Nodes therefore transmit relative humidity, and the server derives and stores absolute humidity on receipt.

```
src/services/iaq/humidity.ts
```

```
TYPESCRIPT
```

```
/**
 * Absolute humidity (g/m³) via the Magnus approximation.
 * Dehumidification algorithms and condensation prediction take this, not relative hu
 */
export function absoluteHumidity(tempC: number, rh: number): number {
  const es = 6.112 * Math.exp((17.67 * tempC) / (tempC + 243.5)); // saturation vapo
  const e = es * (rh / 100); // actual vapor p
```

```

    return (216.7 * e) / (tempC + 273.15); // g/m³
  }

  /** Dew point (°C) - direct input to the condensation-prevention model */
  export function dewPoint(tempC: number, rh: number): number {
    const gamma = (17.67 * tempC) / (tempC + 243.5) + Math.log(rh / 100);
    return (243.5 * gamma) / (17.67 - gamma);
  }

  // absoluteHumidity(20, 60) → 10.35    absoluteHumidity(30, 60) → 18.21

```

Telemetry payload

Nodes **measure and transmit every minute**; the server builds hourly aggregates (TelemetryHourly) and retains raw data for 7 days. Each transmission is signed with the device key; the server confirms the signer address matches `deviceAddress` before looking up the license NFT. Without a signature there is no way to stop arbitrary payloads spoofing an address that holds an NFT.

GET https://api.wellbianlabs.io/v1/c?<base64>

JSON

```

{
  "schema": "wellbian-iaq-telemetry/v1",
  "deviceAddress": "rDeviceWalletAddress...",
  "signature": "3045022100...",
  "batch": [
    {
      "ts": 1786195200,
      "pm25": 11.4, "pm10": 15.8,
      "co2": 812, "tvoc": 143,
      "tempC": 23.4, "rh": 58.1, "feelsLikeC": 24.1
    },
    {
      "ts": 1786195260,
      "pm25": 11.1, "pm10": 15.2,
      "co2": 845, "tvoc": 139,
      "tempC": 23.4, "rh": 58.4, "feelsLikeC": 24.2
    }
  ]
}

```

Receipt validation order

#	CHECK	ON FAILURE	NOTES
1	Signature matches the public key of <code>deviceAddress</code>	401 <code>BAD_SIGNATURE</code>	No on-chain lookup - cheapest check first
2	<code>ts</code> increases monotonically within the batch and is within ± 10 min of server time	422 <code>TIMESTAMP_SKEW</code>	Blocks replay of historical data
3	Each metric is within its physical range	422 <code>OUT_OF_RANGE</code>	First-pass filter for sensor faults and tampering
4	Rate of change vs. previous value is within threshold	Outlier flag	Not rejected - folded into the quality score
5	Holds a license NFT (60-second cache)	403 <code>NO_VALID_LICENSE</code>	Most expensive check last
6	Submission interval floor respected	429 <code>RATE_LIMITED</code>	Prevents reward farming

RAW DATA DOES NOT GO ON-CHAIN

Writing one-minute-resolution measurements times the node count to the ledger is impossible on both cost and performance. Raw data lives in off-chain storage, and **only the batch Merkle root is recorded in the `Memos` field of an XRPL transaction each settlement period**. If the integrity of a specific measurement is later disputed, a Merkle proof verifies that "this value was included in the batch at that time." This is how Whitepaper 3.2's "tamper-proof storage" is achieved.

AQ Spec

Licensing Tiers & Quality Score

Quantifies Whitepaper ch. 5's "high-quality data" and defines rewards purely in quantity, so that total emission is fixed in advance by phase gates and the ten-phase cap (2.665B).

Taxon and tier scheme

The license NFT distinguishes networks via `NFTokenTaxon` and expresses the sensor configuration grade via `license.tier` in the metadata. A tier distinguishes **sensor configuration** and **accessible data products**; the reward quantity is set by phase and quality grade, not by tier.

TAXON	NETWORK	TIER	REQUIRED SENSORS	REWARD
3026	Indoor IAQ	iaq-lite	PM2.5/10, temp/RH	TBD (set at launch)
3026	Indoor IAQ	iaq-standard	PM2.5, PM10, CO ₂ (NDIR), TVOC, temp/RH (ARC-600DA)	Phase-1 base 7/day (normal grade · not guaranteed)
3026	Indoor IAQ	iaq-pro	Additional sensors TBD (set at launch)	Same rule (phase × quality grade)
2026	Outdoor OAQ	oaq-standard	PM2.5/10, O ₃ , NO ₂ , temp/RH	TBD (set when the outdoor model launches)
2026	Outdoor OAQ	oaq-pro	+ SO ₂ , CO, wind speed/direction	TBD (set when the outdoor model launches)

Device metadata (IPFS)

Housing attributes are essential to the "climate-control efficiency by floor area, location, and daylight" analysis flagged in Whitepaper ch. 2. But the combination of these attributes can itself pinpoint a household, so they are **recorded only as buckets - never continuous values - starting at the metadata stage**.

```
ipfs://Qm.../iaq-device.json
```

```
JSON
```

```
{
  "schema": "wellbian-iaq-device/v1",
  "name": "Wellbian IAQ Node #10427",
  "description": "Indoor Air Quality license",
  "device": {
    "serial": "KW-IAQ-10427",
    "model": "ARC-600DA",
```

```

    "sensors": ["pm25", "pm10", "co2", "tvoc", "temp", "rh"],
    "installed_at": "2026-05-02"
  },
  "context": {
    "geohash5": "wydm9",
    "space_type": "residential",
    "area_bucket": "60-85sqm",
    "floor_bucket": "5-10F",
    "build_year_bucket": "2010-2019",
    "orientation": "S"
  },
  "license": { "tier": "iaq-standard", "valid_until": "2029-05-02" }
}

```

METADATA IS PUBLIC

An NFT's `URI` is recorded on-chain and IPFS content is readable by anyone. **Putting an exact address, unit number, or continuous coordinates here publishes them permanently.** Record only `geohash5` (roughly a 4.9 km cell) and bucket values; no precise address is collected or stored - location is retained only at the administrative-dong level, with rounded coordinates.

Quality Score

Decomposes Whitepaper ch. 5's "high-quality data around the clock" into four observable indicators. **Neighbor-node correlation** is the critical one: it catches sensors left outdoors or values synthesized outright - if temperature and PM trends bear no relation to other nodes in the same area, that node's data cannot be trusted.

src/services/iaq/quality.ts

TYPESCRIPT

```

export interface QualityInputs {
  uptimeRatio: number; // actual submissions / expected submissions, last 24h
  outlierRatio: number; // share of out-of-range + sudden-jump outliers
  neighborCorr: number; // temp/PM trend correlation with nodes within 2km (-1 to 1)
  calibrationAgeDays: number; // days since last calibration
}

const clamp01 = (x: number) => Math.min(1, Math.max(0, x));

/** 0 to 1. Weights sum to 1.0 */
export function qualityScore(q: QualityInputs): number {

```

```

const uptime = clamp01(q.uptimeRatio);
if (uptime < 0.2) return 0; // 가동률 20% 미만이면 점수 0
const clean = clamp01(1 - q.outlierRatio * 4); // 25% outliers scores zero
const agree = clamp01((q.neighborCorr - 0.3) / 0.6); // zero below 0.3, full mark
const fresh = clamp01(1 - q.calibrationAgeDays / 365); // zero after a year

const score = 0.40 * uptime + 0.25 * clean + 0.20 * agree + 0.15 * fresh;
return Math.round(score * 1000) / 1000;
}

```

Fixed budget distribution (emission cap)

Paying a value per node makes emission swing with price. Instead the **quantity is fixed** - the epoch budget is `active points × 7.14 × phase factor`, and each node first receives 7 (normal), 8.4 (excellent) or 0 (failing) by quality grade. The unpaid share is redistributed pro rata by weight, capped at 10.5 per point, with the excess burned.

src/services/iaq/reward.ts

TYPESCRIPT

```

import { qualityScore, QualityInputs } from './quality';

export const BASE_DAILY = 7; // 정상 등급 1차 지급 (수량 · 시세 무관)
export const EXCELLENT_DAILY = 8.4; // 품질 상위 10% (+1.4)
export const NODE_CAP = 10.5; // 재분배 후 지점당 상한 - 초과분 소각
export const BUDGET_PER_NODE = 7.14; // 에폭 예산 = 활성 지점 × 7.14 × 단계 계수
export const PHASE_DECAY = 0.78; // 단계마다 22% 체감

export type Grade = 0 | 1 | 1.2;
export interface NodeEpochStats { address: string; quality: QualityInputs; txRate: number }

/** 전송률 게이트 → 품질 점수 상위 10% 우수. 게이트 미달은 0 */
export function grade(n: NodeEpochStats, top10Cut: number): Grade {
  if (n.txRate < 0.8) return 0;
  return qualityScore(n.quality) >= top10Cut ? 1.2 : 1;
}

/** 에폭 정산: 총 지급 + 소각 = 예산 (항상 성립) */
export function distribute(nodes: NodeEpochStats[], phase: number) {
  const factor = Math.pow(PHASE_DECAY, phase - 1);
  const active = nodes.filter((n) => n.licenseHeld);
  const budget = active.length * BUDGET_PER_NODE * factor;
  const cut = percentile(active.map((n) => qualityScore(n.quality)), 0.9);

```

```
// 1) 1차 지급 - 정상 7 · 우수 8.4 · 미달 0 (80~95% 전송률은 선형 감액)
const paid = active.map((n) => {
  const g = grade(n, cut);
  const ramp = n.txRate >= 0.95 ? 1 : n.txRate < 0.8 ? 0 : (n.txRate - 0.8) / 0.15;
  return { address: n.address, g, amount: BASE_DAILY * g * ramp * factor };
});
const unpaid = budget - paid.reduce((s, p) => s + p.amount, 0);

// 2) 미지급분 재분배 - 가중치 비례, 지점당 상한 10.5 × factor, 초과분 소각
const w = paid.reduce((s, p) => s + p.g, 0);
let burn = 0;
for (const p of paid) {
  if (p.g === 0 || w === 0) continue;
  const target = Math.min(p.amount + unpaid * (p.g / w), NODE_CAP * factor);
  burn += p.amount + unpaid * (p.g / w) - target;
  p.amount = target;
}
return { paid, burn, budget }; // Σ paid.amount + burn === budget
}
```

CLOSING THE "REWARD INFLATION" RESIDUAL RISK

This design structurally closes the reward-inflation item in the [residual risks](#) table. Ten-phase cumulative emission is fixed in advance at **2.665B** (within the 4.6B round-1 + round-2 node budget), and the 22% phase decay shrinks the per-point quantity as points grow. Quality shortfalls are burned rather than paid, so **worse quality means less emission**, and every phase gate carries a demand condition, leaving no path where emission grows without demand.

SWITCHED TO BATCH PAYOUT

Fixed budget distribution can only be computed **once every node's score is known at epoch close**. The per-request immediate payout in [Step 4](#) is therefore not used on the IAQ network. Telemetry receipt responds immediately with `202 Accepted`, and rewards from the `distribute()` result accrue in a daily epoch batch closing at KST midnight, and become claimable after a 24-hour hold. This also dovetails naturally with Sequence contention mitigation.

Turns Whitepaper 3.2's "de-identify personal data, then process into metadata" into five concrete transformations. Indoor data is **sensitive information that exposes occupancy and daily routine directly**, so it demands far stronger handling than outdoor weather data.

WHAT INDOOR DATA REVEALS

A CO₂ concentration time series alone estimates **occupancy count, sleep and wake times, and absence periods** with considerable accuracy. PM2.5 spikes reveal cooking times; TVOC patterns reveal cleaning and interior work. Combined with precise location, this data effectively becomes a specific household's activity log. Accordingly, **no path may exist where precise location, raw time resolution, and per-node identifiers leave the platform together**.

Five transformations

#	PHASE	RAW	PUBLIC
P1	Spatial generalization	Dong-level location · rounded coordinates	<code>geohash5</code> - roughly a 4.9 km cell
P2	Temporal aggregation	1-minute resolution measurements	Hourly mean · max · variance
P3	Identifier substitution	<code>rDeviceAddress...</code>	<code>HMAC(NFTokenID, epochSalt)</code> - rotated every epoch
P4	k-anonymity gate	-	Roll up to a coarser cell if the bucket holds < 5 nodes; refuse the response if still short
P5	Attribute generalization	84.3 m ² , 7th floor	<code>60-85sqm</code> , <code>5-10F</code>

P3's **per-epoch salt rotation** matters. Hashing with a fixed salt turns the pseudonym into a permanent identifier and enables long-term tracking. Rotating the salt each epoch severs the link between epochs. The trade-off is that long-term time series analysis becomes impossible, so long-range analysis is served **only as bucket-level aggregates, never via pseudonyms**.

```

export const K_MIN = 5;

export interface NodeSummary { geohash5: string; tier: string; hourly: HourlyStats; }
export interface Bucket { geohash: string; precision: number; n: number; stats: HourlyStats; }

/**
 * P4 - roll up to coarser cells until k-anonymity is satisfied.
 * If geohash3 (~156km) is still short, return null and the API refuses to respond.
 */
export function rollupToKAnonymous(nodes: NodeSummary[], geohash: string): Bucket | null {
  for (let precision = Math.min(5, geohash.length); precision >= 3; precision--) {
    const prefix = geohash.slice(0, precision);
    const members = nodes.filter((n) => n.geohash5.startsWith(prefix));

    if (members.length >= K_MIN) {
      return { geohash: prefix, precision, n: members.length, stats: aggregate(members) };
    }
  }
  return null;
}

/** P3 - derive a pseudonymous ID from a salt rotated each epoch */
export function pseudonym(nftokenId: string, epochSalt: string): string {
  return createHmac('sha256', epochSalt).update(nftokenId).digest('hex').slice(0, 16)
}

```

Differential attack defenses

k-anonymity alone is not enough. Repeating queries with slightly varied filters lets an attacker **isolate individual households from the difference** between two results. If the "60–85 m² & floors 5–10" cohort and the "60–85 m² & floors 5–10 & south-facing" cohort differ by 1, that one household's attributes are exposed outright.

DEFENSE	RULE	EFFECT
Count rounding	Round the returned cohort size to a multiple of 5	Query pairs differing by 1 return the same value
Filter count cap	At most 3 attribute filters applied simultaneously	Blocks narrow cohorts created by combinatorial explosion

DEFENSE	RULE	EFFECT
Query log · budget	Daily query cap per buyer; detection of repeated similar queries	Limits repeated differential attempts at the source
Minimum cohort	Refuse if still under 25 after rounding	A practical floor more conservative than k=5

Data access matrix

DATA	CONTRIBUTOR (OWN)	APPLIANCE MANUFACTURERS	PUBLIC API
Own node raw measurements (limited retention)	Full	No access	No access
Dong-level location · rounded coordinates (no precise address retained)	Full	No access	No access
Hourly cell aggregates	Full	Subscription	Limited (24h delayed)
Cohort counts	-	Per-query billing	No access
Subject contacts · node list	-	No access	No access

THE PLATFORM BROKERS AD DELIVERY

In Whitepaper ch. 6's "targeted advertising fee" model, what reaches the manufacturer is **the cohort size and the ad creative, nothing else**. The subject list is never transferred. The manufacturer submits a request - "show this creative to the 1,240 households meeting condition X" - and the platform delivers through its own app and notification

channels, settling only on lead count. If this boundary breaks, the entire ecosystem is directly exposed to personal data processing risk.

LEGAL REVIEW REQUIRED

This section is technical design, not legal advice. For domestic service, **specialist privacy counsel must review before work begins**: whether indoor environment data constitutes personal information, pseudonymized-data processing requirements, collection and use consent language, the distinction between third-party provision and delegation, and whether reward payouts could be construed as compensated data sales.

AQ Spec

B2B Data Products & Billing Design

Maps Whitepaper ch. 6's three revenue streams onto three products on the x402 protocol. Because billing is unified, on-chain settlement works without a separate billing system.

REVENUE STREAM (WP CH. 6)	ENDPOINT	RESPONSE
API subscription	GET https://wellbian.io/api/data/aggregates	Metric aggregates by cell and time window
Targeted advertising fees	POST https://wellbian.io/api/data/cohort (planned)	Size only of the matching cohort
Insight report	POST https://wellbian.io/api/data/report (planned)	LLM analysis report
(R&D dataset)	POST https://wellbian.io/api/data/dataset (planned)	De-identified training-set

REVENUE STREAM (WP CH. 6)

ENDPOINT

RESPONSE

snapshot
URL

Cohort API - targeting without exposing individuals

Every marketing application in Whitepaper ch. 4 converges on this single endpoint. The key point is that **the response contains no node list**. It returns size and representative statistics only.

POST /api/data/cohort

JSON

```
{
  "region": "wydm",
  "window": { "from": "2026-08-01", "to": "2026-08-31" },
  "filters": [
    { "metric": "ah", "op": "gte", "value": 14.0, "duty": 0.6 },
    { "metric": "area_bucket", "op": "in", "value": ["60-85sqm", "85-135sqm"] }
  ]
}
```

200 OK - monsoon high-humidity households (WP ch.4 dehumidifier scenario)

JSON

```
{
  "cohortSize": 1240,
  "rounding": 5,
  "kAnonymity": { "min": 5, "satisfied": true, "effectiveMin": 25 },
  "profile": {
    "ah_p50": 15.8, "ah_p90": 19.2,
    "hours_above_threshold_p50": 431,
    "top_area_bucket": "60-85sqm"
  },
  "reachable": true,
  "note": "No subject list is provided. Ad delivery is brokered by the platform."
}
```

422 - cohort too narrow

JSON

```
{
  "error": "COHORT_TOO_NARROW",
  "minimum": 25,
}
```

```
"hint": "Relax the filters or reduce region precision. The actual size is not returned"
}
```

Putting subscriptions on x402 - prepaid credit

Vanilla x402 is **one call = one payment**. Applying that literally to a product called dozens of times per second, like the aggregate API, means waiting 3–5 seconds for ledger finality on every call - it simply does not work. The 402 response therefore **offers two schemes at once: per-call payment and prepaid credit**, and subscription customers take the latter.

HTTP/1.1 402 - two payment schemes offered

JSON

```
{
  "x402Version": 1,
  "resource": "/api/data/aggregates",
  "accepts": [
    {
      "scheme": "xrpl-payment",
      "network": "xrpl-mainnet",
      "payTo": "rhTTtx8YyKzcmPDsoroYXuHdGMThrnweb1",
      "asset": { "currency": "RLUSD", "issuer": "rMxCKbEDwqr76QuheSUMdEGf4B9xJ8m5De", "destinationTag": 402010, "maxTimeoutSeconds": 120 },
    },
    {
      "scheme": "xrpl-credit",
      "network": "xrpl-mainnet",
      "payTo": "rhTTtx8YyKzcmPDsoroYXuHdGMThrnweb1",
      "topUpMinimum": { "currency": "RLUSD", "issuer": "rMxCKbEDwqr76QuheSUMdEGf4B9xJ8m5De", "destinationTag": 402011, "unitPrice": "0.002", "authScheme": "x402-credit .." },
    }
  ]
}
```

Topping up credit is a single on-chain Payment; subsequent calls are **authenticated by signature alone**. A nonce is included in the signed message to prevent replay, and nonces are required to increase monotonically.

src/services/x402.credit.ts

TYPESCRIPT

```

import { verify } from 'ripple-keypairs';
import { verifyPayment } from './x402';

interface CreditAccount { balance: number; currency: 'RLUSD' | 'WLBN'; lastNonce: number; }
const credits = new Map<string, CreditAccount>(); // use a persistent store in prod

/** Top-up - reuses the existing on-chain verification logic as-is */
export async function topUp(txHash: string) {
  const res = await verifyPayment(txHash); // replay + delivered-amount check
  if (!res.ok) return res;

  const acc = credits.get(res.payer) ?? { balance: 0, currency: res.currency, lastNonce: 0 };
  acc.balance += Number(res.value);
  credits.set(res.payer, acc);
  return { ok: true as const, balance: acc.balance };
}

/** Call - signature + balance only, no on-chain round trip (~1ms) */
export function spend(header: string, method: string, path: string, unitPrice: number) {
  const [payer, nonceStr, signature] = header.split('.');
  const acc = credits.get(payer);
  if (!acc) return { ok: false as const, reason: 'NO_CREDIT_ACCOUNT' };

  const nonce = Number(nonceStr);
  if (!Number.isInteger(nonce) || nonce <= acc.lastNonce) return { ok: false as const, reason: 'INVALID_NONCE' };

  const message = Buffer.from(`${method}.${path}.${nonce}`, 'utf8').toString('hex').toUpperCase();
  if (!verify(message, signature, acc.publicKey)) return { ok: false as const, reason: 'INVALID_SIGNATURE' };

  if (acc.balance < unitPrice) return { ok: false as const, reason: 'INSUFFICIENT_CREDITS' };

  acc.balance -= unitPrice;
  acc.lastNonce = nonce;
  return { ok: true as const, remaining: acc.balance };
}

```

HOW THE TWO SCHEMES DIVIDE THE WORK

`xrpl-payment` serves **low-frequency, high-value** products (cohort queries, insight reports); `xrpl-credit` serves **high-frequency, low-value** ones (the aggregate API).

When the credit balance runs out the server returns 402 again, so from the buyer agent's perspective top-up and usage are handled within the same protocol flow.

Reward spending and payment structure

The "ecosystem spending" of Whitepaper ch. 5 (appliance discounts, filter subscriptions, partner marketplace) will be specified separately when the partner marketplace opens. Payments follow the published policy - enterprise customers pay in KRW or RLUSD, AI agents pay automatically (x402), and the protocol buys WLBN with part of revenue and burns it. Customers are never asked to buy the token to pay, and allocation ratios are announced in advance.

Value-flow summary

fund flow over one epoch

LOG

```
[IN ] Enterprise → Hot      RLUSD / KRW (aggregate credit · cohort · report
[IN ] AI agent   → Hot      RLUSD / WLBN (per-call x402 payment)
[OUT] Hot        → points   WLBN 7 × 0.78^(phase-1) × grade ← quantity,
[BURN] Hot       → Issuer   WLBN quality shortfall (redistribution excess)
[BURN] Treasury → Issuer   WLBN scheduled 2.5B in 4 tranches · buyback fr
[LP ] Treasury  → AMM      WLBN/RLUSD 20% of participation fees on sale
```

STEP 01

XRPL Foundation & Wallet Setup

Prepare the Mainnet client module ([wss://xrplcluster.com](https://xrplcluster.com)) and three wallets (cold Issuer · Hot · Treasury), pre-funding each account with the XRP it needs for reserves.

DELIVERABLES

- [xrpl/client.ts](#) - reusable connection singleton
- [xrpl/wallet.ts](#) - seed-based load + new creation/funding
- [scripts/01-bootstrap-wallets.ts](#) - prints values to paste into [.env](#)

src/xrpl/client.ts

TYPESCRIPT

```
import { Client } from 'xrpl';
import { env } from '../config/env';

let client: Client | null = null;

export async function getClient(): Promise<Client> {
  if (client?.isConnected()) return client;
  client = new Client(env.XRPL_ENDPOINT, { connectionTimeout: 10_000 });
  client.on('disconnected', (code) => console.warn('[xrpl] disconnected', code));
  await client.connect();
  return client;
}

export async function closeClient(): Promise<void> {
  if (client?.isConnected()) await client.disconnect();
  client = null;
}
```

src/scripts/01-bootstrap-wallets.ts

TYPESCRIPT

```
import { Wallet } from 'xrpl';
import { getClient, closeClient } from '../xrpl/client';
import { env } from '../config/env';

const ROLES = ['ISSUER', 'HOT', 'TREASURY'] as const;
const MIN_RESERVE_XRP = 5;
// ISSUER(cold) rDJz8WJhsKgydqJzSZXMpsJot3eRmSkR5
// HOT          rhTTtx8YyKzcmPDsoroYXuHdGMThrnweb1
// TREASURY     rPZFY2yLgqjmBrxtxWHXnB1doBt7gu7xin

async function main() {
  const client = await getClient();

  for (const role of ROLES) {
    // Mainnet: 시드로 로드하고 잔액을 확인한다 (Faucet 없음)
    const wallet = Wallet.fromSeed(env[`_${role}_SEED`]);
    const info = await client.request({
      command: 'account_info', account: wallet.classicAddress, ledger_index: 'validated'
    });
    const balance = Number(info.result.account_data.Balance) / 1_000_000;
    if (balance < MIN_RESERVE_XRP) throw new Error(`${role} underfunded: ${balance} XRP`);
    console.log(`\n# ${role}`);
    // 시드는 출력하지 않는다 - .env에서만 관리한다
    console.log(`${role}_ADDRESS=${wallet.classicAddress} # balance ${balance} XRP`);
  }
}
```

```

    }

    await closeClient();
  }

  main().catch((e) => { console.error(e); process.exit(1); });

```

RESERVE ARITHMETIC

Each account locks a 1 XRP Base Reserve, plus 0.2 XRP for every TrustLine, NFT Offer, or AMM object. The Hot wallet needs an extra 0.4 XRP for its two TrustLines (WLBN + RLUSD), so pre-fund every account with at least 5 XRP and **add a pre-flight balance check when registering devices at scale.**

STEP 02

Wellbian Token (WLBN) Issuance (Issued Asset)

The cold Issuer issues Wellbian (WLBN); holder wallets set up TrustLines for WLBN and for real RLUSD.

Issuance procedure

1. **Issuer account setup** - enable `asfDefaultRipple` via `AccountSet`, optionally set `TransferRate`
2. **Holder TrustLines** - Hot, Treasury, and user wallets set limits via `TrustSet`
3. **Initial issuance** - sending a `Payment` from Issuer → Treasury/Hot brings that amount of the token into existence (IOUs have no separate mint transaction)

DEFAULT RIPPLE IS ENABLED ON THE ISSUER

On XRPL, IOUs always move through the issuer account. The issuer account must therefore set `asfDefaultRipple` **ON**. With this flag off, holder-to-holder transfers (Hot → user wallet) fail with `tecPATH_DRY`, and neither Step 4's reward payout nor Step 5's AMM can work.

Prevent unintended liquidity transfer by applying `tfSetNoRipple` **on holder-side TrustLines**. This combination permits "normal transfers routed through the issuer" while

blocking "third-party paths that use a holder account as a bridge".

src/xrpl/token.ts

TYPESCRIPT

```
import { Client, Wallet, AccountSetAsfFlags, TrustSetFlags, IssuedCurrencyAmount } from 'xrpl';
import { submit } from './submit';

/** Issuer account setup - DefaultRipple ON so IOUs can move between holders */
export async function configureIssuer(client: Client, issuer: Wallet) {
  await submit(client, issuer, {
    TransactionType: 'AccountSet',
    Account: issuer.classicAddress,
    SetFlag: AccountSetAsfFlags.asfDefaultRipple,
    Domain: Buffer.from('wellbian.io').toString('hex').toUpperCase(),
  });
}

/** Holder TrustLine - NoRipple blocks third-party bridge routing */
export async function setTrustLine(
  client: Client, holder: Wallet, issuerAddress: string,
  currency: string, limit = '1000000000',
) {
  await submit(client, holder, {
    TransactionType: 'TrustSet',
    Account: holder.classicAddress,
    LimitAmount: { currency, issuer: issuerAddress, value: limit },
    Flags: TrustSetFlags.tfSetNoRipple,
  });
}

/** IOU transfer - from the Issuer it is issuance; between holders it is circulation */
export async function sendIOU(
  client: Client, from: Wallet, destination: string,
  amount: IssuedCurrencyAmount, destinationTag?: number,
) {
  return submit(client, from, {
    TransactionType: 'Payment',
    Account: from.classicAddress,
    Destination: destination,
    Amount: amount,
    ...(destinationTag !== undefined ? { DestinationTag: destinationTag } : {}),
  });
}
```

Token definitions

TOKEN	CURRENCY (HEX)	ROLE	INITIAL SUPPLY
WLBN	574C424E0000...	DePIN rewards · API fee settlement	10,000,000,000 (total issued)
RLUSD	524C5553440000...	Stable pair · inbound currency for institutional payment (Issuer rMxCkEDwqr76QuheSUMdEGf4B9xJ8m5De)	Externally issued (Ripple) - not self-issued

Supply Total issuance is **10 billion WLBN**. XRPL IOUs have no total-supply field, so supply is defined as the sum of obligations the Issuer has extended (obligations in `gateway_balances`). As of 2026-09-01 the distribution is 9,999.95M in the treasury, about 50K in the hot wallet (reward fund), and about 5 in circulation.

Emission schedule - phase decay

Total issuance is 10B, but it is **never released at once**. The emission baseline is **7 WLBN per point per day · 213 per month · 2,555 per year** (normal grade) in indoor Phase 1, decaying 22% per phase ($7 \times 0.78^{(\text{phase}-1)}$). These are planning figures, not promised payouts; actual receipts depend on quality grade and redistribution.

PHASE	DAILY BASE	VS. PHASE 1	PHASE EMISSION
1 Domestic indoor (25,000 points)	7.00	100%	64M
3	4.26	61%	187M
5	2.59	37%	331M
7	1.58	23%	374M
10	0.75	11%	273M
Ten-phase total			2.665B (within the 4.6B round-1 + round-2 node budget)

Target scale Phase 1 is **25,000 domestic indoor points** (genesis-only recruitment). Expansion then goes indoor → outdoor, with the outdoor line offered first to existing

holders, toward a final target of a **1M-point** integrated network. Phases open only through [milestone gates](#) that require both point count and demand.

Allocation Of the 10B total, node-holder rewards are **1B (10%) in round 1** and **3.6B (36%) in round 2**, which opens as phase gates are passed after round 1 is spent. Scheduled burn 1.5B, strategic round reserve 1B, staking pool 0.8B, team 0.6B, ecosystem 0.5B, strategic 0.5B, liquidity & reserve 0.5B - the full table is in the [tokenomics policy](#) section.

Decay A time-based halving is **not used**; halving every year would unsettle the planned emission baseline. Instead emission **decays 22% per phase**, phases being defined by point-count gates, and the brake is the gate (demand condition) and the ten-phase cap rather than time.

Cap The first-pass payout per point is a fixed quantity by grade (7 normal · 8.4 excellent · 0 failing), and even after redistribution no point exceeds **10.5**. The excess is burned rather than paid, so paid + burned always equals the epoch budget.

Note Real RLUSD is a stablecoin issued by Ripple. Service charges (x402) are collected only in this RLUSD, and the issuer address is pinned as a constant: **rMxCKbEDwqr76QuheSUMdEGf4B9xJ8m5De**. No self-issued mock RLUSD is used.

STEP 03

IoT License NFT (XLS-20)

One device = one NFT. Data submission rights are decided by querying license ownership on-chain.

Redeem in four steps - (1) On pre-order or promotion a code is issued **UNBOUND** (**WLBN-XXXX-XXXX-XXXX**). (2) The user enters the code plus the device serial at wellbian.io/redeem, binding the code to that device. (3) The Hot wallet mints the NFT (Issuer is the cold wallet) and creates an **NFTokenCreateOffer** with Amount=0 to the user address. (4) The user confirms receipt with **NFTokenAcceptOffer**. Supported wallets: embedded wallet (ID/password or Google sign-in), D'CENT mobile in-app, Xaman, GemWallet, and Crossmark.

Mint → transfer flow

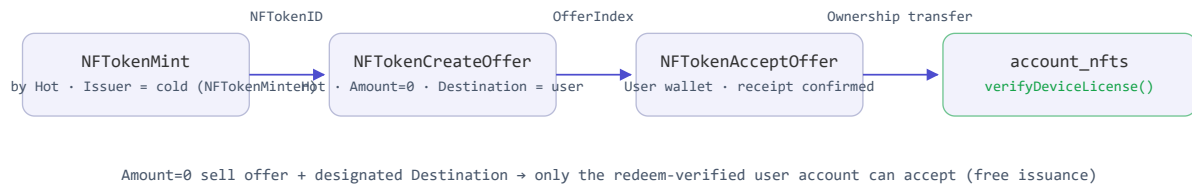


Fig 2. XLS-20 license issuance sequence

src/xrpl/nft.ts

TYPESCRIPT

```

import { Client, Wallet, NFTokenMintFlags, NFTokenCreateOfferFlags, convertStringToHex } from 'xrpl';
import { submit } from './submit';

export const TAXON = {
  OBS_GENESIS: 1001, OBS_REGION: 1002, OBS_BIZ: 1003, // 관측소 NFT (계정당 1개)
  VOUCHER: 1010, // 사전예약 바우처 (테스트용)
  OAQ: 2026, IAQ: 3026, AI: 4026, RADAR: 5026, SAT: 6026, // 스테이션 NFT (기기당 1개)
} as const;

/** Mint a license NFT → returns the NFTokenID */
export async function mintLicense(
  client: Client, hot: Wallet, metadataUri: string, taxon: number = TAXON.IAQ,
) {
  // 서명자는 Hot 지갑, 발행 주체(Issuer)는 콜드 지갑 - NFTokenMinter로 위임
  const res = await submit(client, hot, {
    TransactionType: 'NFTokenMint',
    Account: hot.classicAddress,
    Issuer: env.ISSUER_ADDRESS,
    URI: convertStringToHex(metadataUri), // ipfs://Qm...
    NFTokenTaxon: taxon,
    Flags: NFTokenMintFlags.tfTransferable,
  });

  // meta.nftoken_id is populated by rippled
  const meta = res.result.meta as { nftoken_id?: string };
  if (!meta.nftoken_id) throw new Error('NFTokenID not found in metadata');
  return meta.nftoken_id;
}

/** Create a zero-price sell offer restricted to one recipient */
export async function offerLicenseTo(
  client: Client, hot: Wallet, nftokenId: string, destination: string,
) {

```

```

const res = await submit(client, hot, {
  TransactionType: 'NFTokenCreateOffer',
  Account: hot.classicAddress,
  NFTokenID: nftokenId,
  Amount: '0',
  Destination: destination, // 리딤으로 확인된 사용자 계정만 수락 가능
  Flags: NFTokenCreateOfferFlags.tfSellNFToken,
});

const meta = res.result.meta as { offer_id?: string };
if (!meta.offer_id) throw new Error('OfferID not found in metadata');
return meta.offer_id;
}

/** User wallet accepts the offer → ownership transfers.
 * D'CENT 펜딩 오퍼 풀백은 설정 플래그 allowPendingOffer(기본 false)로 제어한다. */
export async function acceptLicense(client: Client, user: Wallet, offerId: string) {
  return submit(client, user, {
    TransactionType: 'NFTokenAcceptOffer',
    Account: user.classicAddress,
    NFTokenSellOffer: offerId,
  });
}

```

On-chain license verification

The core function the oracle endpoint calls on every request. **Checking the Issuer address together with the Taxon** is what filters out forged NFTs minted by arbitrary third parties.

src/xrpl/nft.ts (continued)

TYPESCRIPT

```

import { env } from '../config/env';

export interface LicenseInfo {
  valid: boolean;
  nftokenId?: string;
  uri?: string;
  taxon?: number;
}

/** Verify on-chain that the address holds a valid license NFT for this network */
export async function verifyDeviceLicense(

```

```

    client: Client, walletAddress: string, taxon: number = TAXON.IAQ,
  ): Promise<LicenseInfo> {
    const res = await client.request({
      command: 'account_nfts',
      account: walletAddress,
      ledger_index: 'validated',
    });

    const hit = res.result.account_nfts.find(
      (n) => n.Issuer === env.ISSUER_ADDRESS && n.NFTTokenTaxon === taxon,
    );
    if (!hit) return { valid: false };

    return {
      valid: true,
      nftokenId: hit.NFTTokenID,
      taxon: hit.NFTTokenTaxon,
      uri: hit.URI ? Buffer.from(hit.URI, 'hex').toString('utf8') : undefined,
    };
  }

```

METADATA SCHEMA (IPFS) - OUTDOOR OAQ EXAMPLE

ipfs://Qm.../oaq-device.json

JSON

```

{
  "schema": "wellbian-oaq-device/v1",
  "name": "Wellbian OAQ Node #0042",
  "description": "Outdoor Air Quality station license",
  "device": {
    "serial": "KW-OAQ-0042",
    "model": "TBD",
    "sensors": ["pm25", "pm10", "o3", "no2", "temp", "rh"],
    "installed_at": "2026-03-11"
  },
  "context": { "geohash5": "wydm9", "space_type": "outdoor" },
  "license": { "tier": "oaq-standard", "valid_until": "2029-03-11" }
}

```

Indoor IAQ node metadata additionally carries housing environment buckets - see the [wellbian-iaq-device/v1](#) schema in [Licensing Tiers & Quality Score](#). Both schemas share the rule that **precise coordinates are never included**.

CACHING AND REVOCATION

Calling `account_nfts` on every request ties telemetry throughput to ledger lookups.

Cache the result per address with a 30–60 second TTL, and define a revocation policy alongside it: reclaim the NFT (re-offer), or mint with `tfBurnable` and have the cold Issuer that holds minting authority burn it via `NFTokenBurn`. Note that with `tfTransferable` alone the issuer cannot force a reclaim.

STEP 04

DePIN Oracle & Reward Engine

An unattended pipeline from telemetry receipt through license verification to WLBN reward payout.

API specification

ITEM	DETAIL
Endpoint	<code>GET https://api.wellbianlabs.io/v1/c?<base64></code> (Supabase Edge Function proxy)
Body	<code>{ schema, deviceAddress, token, batch: [{ ts, pm25, co2, tvoc, tempC, rh, ... }] }</code>
202	<code>{ accepted: 5 }</code> - receipt acknowledged. Rewards accrue in the once-daily epoch settlement and are paid on claim after a 24-hour hold
401	<code>{ error: "BAD_TOKEN" }</code> - collection token mismatch (migrating to per-device tokens; payload signature verification is on the roadmap)
403	<code>{ error: "NO_VALID_LICENSE" }</code> - no license NFT held
422	<code>{ error: "OUT_OF_RANGE" }</code> - a device clock off by more than 10 minutes is not rejected; the reading is corrected to server receive time (KST) and accepted
429	<code>{ error: "RATE_LIMITED" }</code> - submission interval violated

```
src/services/oracle.ts
```

```
TYPESCRIPT
```

```

import { getClient } from '../xrpl/client';
import { verifyDeviceLicense } from '../xrpl/nft';
import { env } from '../config/env';

import { absoluteHumidity } from './iaq/humidity';
import { TAXON } from '../xrpl/nft';

export interface IaqReading {
  ts: number;
  pm25: number; pm10: number;
  co2: number; tvoc: number;
  tempC: number; rh: number;
}

const RANGES: Record<keyof IaqReading, [number, number]> = {
  ts: [0, Number.MAX_SAFE_INTEGER],
  pm25: [0, 1000], pm10: [0, 1000],
  co2: [350, 5000], tvoc: [0, 60000],
  tempC: [-10, 60], rh: [0, 100],
};

const lastSeen = new Map<string, number>();
const MIN_INTERVAL_MS = 4 * 60_000; // 1-minute transmit cadence; floor keeps sl

export async function ingest(deviceAddress: string, batch: IaqReading[]) {
  // 1) submission interval floor - prevents reward farming
  const prev = lastSeen.get(deviceAddress) ?? 0;
  if (Date.now() - prev < MIN_INTERVAL_MS) throw new HttpError(429, 'RATE_LIMITED');

  // 2) physical plausibility - check the whole batch
  for (const reading of batch) {
    for (const [k, [lo, hi]] of Object.entries(RANGES)) {
      const v = reading[k as keyof IaqReading];
      if (typeof v !== 'number' || v < lo || v > hi) throw new HttpError(422, 'OUT_OF');
    }
  }

  // 3) on-chain license check (IAQ Taxon, 60s TTL cache)
  const client = await getClient();
  const license = await verifyDeviceLicense(client, deviceAddress, TAXON.IAQ);
  if (!license.valid) throw new HttpError(403, 'NO_VALID_LICENSE');

  // 4) derive absolute humidity, store off-chain - only the batch Merkle root goes c
  const enriched = batch.map((r) => ({ ...r, ah: absoluteHumidity(r.tempC, r.rh) }));

```

```
await saveTelemetry(deviceAddress, enriched, license.nftokenId!);

// 5) accrual only - the daily epoch settles accruals; payout happens on claim
await accrueForEpoch(deviceAddress, license, enriched);

lastSeen.set(deviceAddress, Date.now());
return { accepted: batch.length }; // HTTP 202
}
```

Reward engine parameters

ITEM	POLICY VALUE	CURRENT IMPLEMENTATION (SEPT 2026)
Epoch	1 day · KST midnight boundary	Same
Daily reward	$7 \times 0.78^{(\text{phase}-1)} \times \text{quality grade}$ (0 · 1.0 · 1.2)	Phase-1 normal 7, flat accrual
Epoch budget	active points × 7.14 × phase factor	Equivalent via the 7-per-point cap
Redistribution	Unpaid share pro rata · cap 10.5 per point · excess burned	To be enabled as a governance variable
Quality grade	Transmission 95%+ full · 80–95% linear · below 80% zero · top 10% by quality score excellent	Quality score (uptime 40 · outlier 25 · correlation 20 · calibration 15) computed · 0 if uptime < 0.2
Claim hold	24 hours	Same
Emission cap	2.665B over ten phases (within the 4.6B round-1 + round-2 node budget)	Accrual stops when the reward pool is exhausted

THE IAQ NETWORK USES BATCH SETTLEMENT

Emitting one Payment per request puts a **3–5 second ledger finality wait** inside the HTTP response, and Sequence contention worsens as nodes multiply. More fundamentally, [fixed budget distribution](#) can only be computed once every node's quality score is known at epoch close, so immediate payout does not work at all.

Receipt therefore responds immediately with `202 Accepted`, and an epoch is a fixed one-day unit bounded by KST midnight. Settlement runs on a resumable five-minute step-runner cron, and hot-wallet transactions are emitted through an outbox pattern. Rewards accrue; payout happens at claim time. The reward tier for the outdoor OAQ network is not yet defined.

SEQUENCE CONTENTION

When a single Hot Wallet signs several Payments concurrently, `Sequence` numbers collide and produce `tefPAST_SEQ` / `terPRE_SEQ`. Hot Wallet submissions are recorded through an **outbox pattern** and drained sequentially by a single worker, resuming from the incomplete outbox if the cron stops.

STEP 05

AMM Liquidity Pool (XLS-30)

Creates a WLBN ↔ RLUSD pool so instant swaps happen within the ledger. The AMM exists solely for swap convenience and performs no price discovery or valuation of Wellbian (WLBN).

AMMCREATE

Pool creation

The initial deposit ratio of the two assets sets the swap exchange rate. A special fee (= Owner Reserve, 0.2 XRP on mainnet) applies.

AMMDEPOSIT

Liquidity provision

Single-asset or two-asset deposits are both supported. You receive LPTokens.

PAYMENT + PATHS

Swap execution

There is no separate swap transaction. A cross-currency Payment routes through the AMM and order book automatically.

src/xrpl/amm.ts

TYPESCRIPT

```
import { Client, Wallet, AMMDepositFlags } from 'xrpl';
import { submit } from './submit';
import { WLBN, RLUSD } from '../config/currency';
import { env } from '../config/env';

const wlbn = (value: string) => ({ currency: WLBN, issuer: env.ISSUER_ADDRESS, value });
const rlusd = (value: string) => ({ currency: RLUSD, issuer: env.RLUSD_ISSUER_ADDRESS, value });

/** AMM pool deposit 100,000 WLBN : 1,000 RLUSD - 예시 값. 예치 비율은 스왑 교환비 산정용임
 * WLBN의 가치·법정화폐 환산을 의미하지 않는다. */
export async function createAmm(client: Client, lp: Wallet) {
  return submit(client, lp, {
    TransactionType: 'AMMCreate',
    Account: lp.classicAddress,
    Amount: wlbn('100000'),
    Amount2: rlusd('1000'),
    TradingFee: 500, // 0.5% (unit: 1/100,000)
  });
}

/** Two-asset deposit */
export async function depositBoth(client: Client, lp: Wallet, a: string, b: string) {
  return submit(client, lp, {
    TransactionType: 'AMMDeposit',
    Account: lp.classicAddress,
    Asset: { currency: WLBN, issuer: env.ISSUER_ADDRESS },
    Asset2: { currency: RLUSD, issuer: env.RLUSD_ISSUER_ADDRESS },
    Amount: wlbn(a),
    Amount2: rlusd(b),
    Flags: AMMDepositFlags.tfTwoAsset,
  });
}

/** RLUSD → WLBN swap: fixed target amount + max spend cap (slippage defense) */
export async function swapRlusdForWlbn(
  client: Client, buyer: Wallet, targetWlbn: string, maxRlusd: string,
) {
  const paths = await client.request({
    command: 'path_find', subcommand: 'create',
  });
}
```

```

    source_account: buyer.classicAddress,
    destination_account: buyer.classicAddress,
    destination_amount: wlnb(targetWlnb),
    source_currencies: [{ currency: RLUSD, issuer: env.RLUSD_ISSUER_ADDRESS }],
  } as never);

return submit(client, buyer, {
  TransactionType: 'Payment',
  Account: buyer.classicAddress,
  Destination: buyer.classicAddress,    // to yourself = currency conversion
  Amount: wlnb(targetWlnb),
  SendMax: rlusd(maxRlusd),             // slippage cap
  Paths: (paths.result as { alternatives: { paths_computed: unknown }[] })
    .alternatives[0]?.paths_computed as never,
  Flags: 0,                             // tfPartialPayment 미사용
});
}

/** Query pool state */
export async function getPoolState(client: Client) {
  const res = await client.request({
    command: 'amm_info',
    asset: { currency: WBNB, issuer: env.ISSUER_ADDRESS },
    asset2: { currency: RLUSD, issuer: env.RLUSD_ISSUER_ADDRESS },
  } as never);
  return res.result;
}

```

SWAP CAVEATS

- **A Payment sent to yourself** is XRPL's currency-conversion idiom. Set `Destination` to your own address.
- Always specify `SendMax`. Without it, pool conditions can consume far more RLUSD than expected.
- With `tfPartialPayment` enabled, a shortfall still counts as success, so payment verification must check the **actual delivered amount** (`delivered_amount`). That is why Step 6's verification uses this field.
- Both assets must have TrustLines set up on the buyer's account.

STEP 06

x402 Autonomous Payment Gateway

The autonomous payment gateway running on mainnet. An external AI agent reads the HTTP 402 response with no human involvement, pays on-chain, and retries to obtain the data.

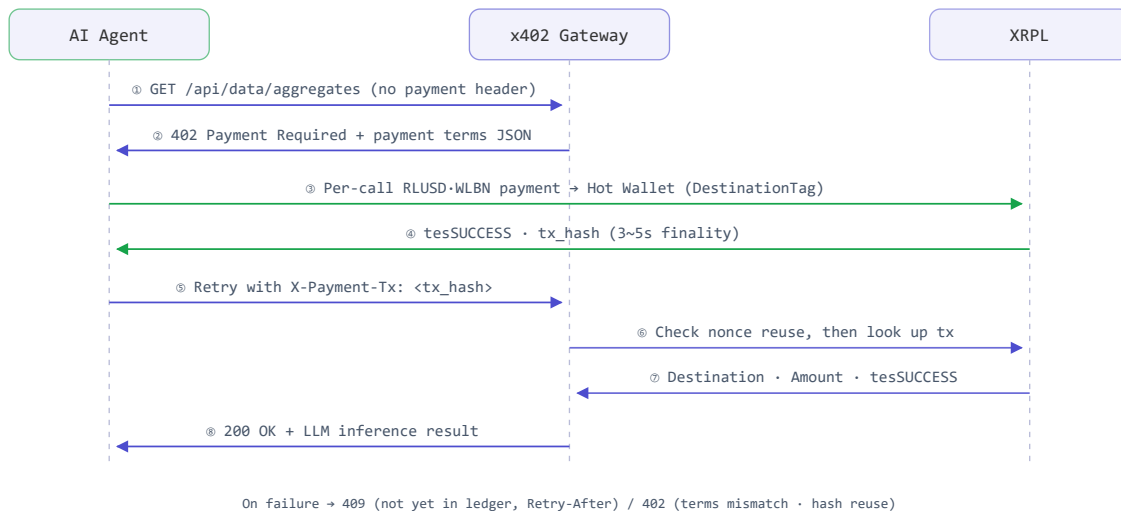


Fig 3. x402 challenge-payment-verification sequence

First request - the 402 challenge

HTTP/1.1 402 Payment Required

HTTP

```

HTTP/1.1 402 Payment Required
Content-Type: application/json
WWW-Authenticate: x402 realm="wellbian-data", network="xrpl-mainnet"

{
  "x402Version": 1,
  "resource": "/api/data/aggregates",
  "accepts": [
    {
      "scheme": "xrpl-payment",
      "network": "xrpl-mainnet",
      "payTo": "rhTTtx8YyKzcmPDsoroYXuHdGMThrnweb1",
      "asset": { "currency": "WLBN", "issuer": "rDJz8WJhsKgydqJzSZXMpsJot3eRmSkR5", "
      "destinationTag": 402001,
      "maxTimeoutSeconds": 120
    }
  ],
}

```

```

    {
      "scheme": "xrpl-payment",
      "network": "xrpl-mainnet",
      "payTo": "rhTTtx8YyKzcmPDsoroYXuHdGMThrnweb1",
      "asset": { "currency": "RLUSD", "issuer": "rMxCKbEDwqr76QuheSUMdEGf4B9xJ8m5De",
      "destinationTag": 402001,
      "maxTimeoutSeconds": 120
    }
  ],
  "description": "Wellbian IAQ 집계 데이터 - 호출당 과금"
}

```

Second request - transaction verification

src/services/x402.ts

TYPESCRIPT

```

import { getClient } from '../xrpl/client';
import { WLBN, RLUSD } from '../config/currency';
import { env } from '../config/env';

const usedHashes = new Set<string>(); // replace with Redis SETNX + TTL in production

export type VerifyResult =
  | { ok: true; currency: 'WLBN' | 'RLUSD'; value: string }
  | { ok: false; reason: string };

const PRICES = {
  [WLBN]: { label: 'WLBN' as const, min: Number(env.X402_PRICE_WLBN) },
  [RLUSD]: { label: 'RLUSD' as const, min: Number(env.X402_PRICE_RLUSD) },
};

export async function verifyPayment(txHash: string): Promise<VerifyResult> {
  // 0) block reuse - stops unlimited calls with the same hash
  if (usedHashes.has(txHash)) return { ok: false, reason: 'HASH_ALREADY_USED' };

  const client = await getClient();
  let res;
  try {
    res = await client.request({ command: 'tx', transaction: txHash, binary: false });
  } catch {
    return { ok: false, reason: 'TX_NOT_FOUND' }; // may not have propagated yet →
  }
}

```

```

const tx = res.result.tx_json ?? (res.result as never as Record<string, unknown>);
const meta = res.result.meta as { TransactionResult: string; delivered_amount?: unknown; }

// 1) Ledger finality + success
if (res.result.validated !== true) return { ok: false, reason: 'NOT_VALIDATED' };
if (meta.TransactionResult !== 'tesSUCCESS') return { ok: false, reason: meta.TransactionResult };
if ((tx as { TransactionType: string }).TransactionType !== 'Payment')
  return { ok: false, reason: 'NOT_A_PAYMENT' };

// 2) recipient matches
if ((tx as { Destination: string }).Destination !== env.HOT_WALLET_ADDRESS)
  return { ok: false, reason: 'WRONG_DESTINATION' };

// 3) check actual delivered amount - trust delivered_amount to block PartialPayment
const delivered = meta.delivered_amount as
  { currency: string; issuer: string; value: string } | string | undefined;
if (!delivered || typeof delivered === 'string')
  return { ok: false, reason: 'XRP_NOT_ACCEPTED' };

const price = PRICES[delivered.currency];
if (!price) return { ok: false, reason: 'UNSUPPORTED_CURRENCY' };
const EXPECTED_ISSUER = { [WLBN]: env.ISSUER_ADDRESS, [RLUSD]: env.RLUSD_ISSUER_ADDRESS };
if (delivered.issuer !== EXPECTED_ISSUER[delivered.currency])
  return { ok: false, reason: 'WRONG_ISSUER' };
if (Number(delivered.value) < price.min) return { ok: false, reason: 'INSUFFICIENT_AMOUNT' };

usedHashes.add(txHash);
return { ok: true, currency: price.label, value: delivered.value };
}

```

Gateway middleware

src/api/middleware.x402.ts

TYPESCRIPT

```

import { RequestHandler } from 'express';
import { verifyPayment } from '../services/x402';
import { buildChallenge } from '../services/x402.challenge';

export const x402Gate: RequestHandler = async (req, res, next) => {
  const payTx = req.header('x-payment-tx'); // 선물 크레딧은 x-api-key 경로로
  const match = payTx?.match(/^[A-F0-9]{64}$/i);

  if (!match) {

```

```

    res.setHeader('WWW-Authenticate', 'x402 realm="wellbian-data", network="xrpl-mainnet"');
    return res.status(402).json(buildChallenge(req.path));
  }

  const result = await verifyPayment(match[1]);
  if (!result.ok) {
    // not yet in the ledger → prompt retry; otherwise reissue 402
    const retryable = result.reason === 'TX_NOT_FOUND' || result.reason === 'NOT_VALID';
    if (retryable) { res.setHeader('Retry-After', '3'); return res.status(409).json({ error: result.reason }); }
    return res.status(402).json({ ...buildChallenge(req.path), error: result.reason });
  }

  res.setHeader('X-Payment-Response', JSON.stringify({ settled: true, ...result }));
  next();
};

```

AI agent client example

src/scripts/agent-example.ts

TYPESCRIPT

```

import { getClient } from '../xrpl/client';
import { sendIOU } from '../xrpl/token';
import { agentWallet, env } from '../config/env';
import { toCurrencyHex } from '../config/currency';

const API = 'https://wellbian.io/api/data/aggregates';
const query = new URLSearchParams({ taxon: 'IAQ', geohash: 'wydm6', from: '2026-08-31' });

async function call(headers: Record<string, string> = {}) {
  return fetch(`${API}?${query}`, { method: 'GET', headers });
}

async function main() {
  // (1) call without payment → receive 402 + payment terms
  const challenge = await call();
  if (challenge.status !== 402) throw new Error(`expected 402, got ${challenge.status}`);
  const terms = await challenge.json();
  const option = terms.accepts.find((a: { asset: { currency: string } }) => a.asset.currency === 'USD');
  console.log('[agent] terms received:', option.asset.value, option.asset.currency);

  // (2) pay on-chain - the agent decides and pays on its own
  const client = await getClient();
  const tx = await sendIOU(

```

```

    client, agentWallet, option.payTo,
    { currency: toCurrencyHex(option.asset.currency), issuer: option.asset.issuer, va
    option.destinationTag,
  });
  const hash = tx.result.hash;
  console.log('[agent] payment settled:', hash);

  // (3) retry with tx_hash attached (retries absorb propagation delay)
  for (let i = 0; i < 5; i++) {
    const res = await call({ 'x-payment-tx': hash });
    if (res.status === 200) { console.log('[agent] response:', await res.json()); ret
    if (res.status !== 409) throw new Error(`failed ${res.status}: ${await res.text()
    await new Promise((r) => setTimeout(r, 3000));
  }
  throw new Error('verification timed out');
}

main().catch((e) => { console.error(e); process.exit(1); });

```

Burn policy - 15% scheduled · revenue-linked buyback burn · published monthly

The scheduled burn is 15% of total supply (1.5B); its tranche schedule will be announced separately. Buyback burns are linked to revenue - 10% of device sales, 10% of AI Factory access pass sales and 50% of data revenue buy WLBN on the market and burn it. Quality-shortfall excess and round-2 rewards left unopened at an unpassed gate are added. Burning returns the IOU to the Issuer; burns are executed monthly and the bought quantity, price and transactions are published.

src/services/burn.ts

TYPESCRIPT

```

import { getClient } from '../xrpl/client';
import { sendIOU } from '../xrpl/token';
import { WLBN } from '../config/currency';
import { env, hotWallet, treasuryWallet } from '../config/env';

export type BurnReason = 'scheduled' | 'buyback' | 'quality-shortfall' | 'round2-unop

/** Return WLBN to the Issuer = burn. Quantity and tx hash are published; buyback tim
export async function burn(amountWlbn: string, reason: BurnReason) {
  if (Number(amountWlbn) <= 0) return { burned: '0' };
  const from = reason === 'quality-shortfall' ? hotWallet : treasuryWallet;

```

```
const client = await getClient();
const res = await sendIOU(client, from, env.ISSUER_ADDRESS, {
  currency: WLBN, issuer: env.ISSUER_ADDRESS, value: amountWlbn,
});
await publishBurn({ reason, amount: amountWlbn, tx: res.result.hash }); // 수량·tx
return { burned: amountWlbn, txHash: res.result.hash };
}
```

"BURNING" ON XRPL IS NOT A SMART CONTRACT BURN

An IOU is a record of the issuer's liability. When a holder sends it back to the issuer, the TrustLine balance is offset and **circulating supply genuinely falls**. No separate burn function is needed. That said, the Issuer account still holds `DefaultRipple` and issuance authority, so guaranteeing "cannot be re-issued" is stronger with a dedicated **blackhole account** (set the `regular key` to `rrrrrrrrrrrrrrrrrrrrrhoLvTp` and disable the master key). The system currently uses offset-by-return-to-Issuer; any move to a blackhole account will be announced separately.

Operations

Tokenomics Policy

Fixed-quantity rewards · 10B total · 15% scheduled burn + revenue-linked buyback burn. This section mirrors the policy published at tokenomics.wellbianlabs.io: it defines quantities only, never value.

On-chain Payout - Pilot Payouts and the DEX Pool Opening

Principle Accrued WLBN is settled daily into the site account and, when claimed, is **paid out on-chain**. In August-September 2026 a **pilot payout (8 claims, 78.23 WLBN)** was executed on-chain for early nodes, and those records remain on the XRPL.

Rationale A market price forms only after the XRPL DEX (AMM) pool opens. Payouts before the pool were limited to small pilot amounts so that no off-market price could form while circulating supply was minimal. The pool opens on **October 2, 2026**.

Guarantee Accrual never stops and **accumulated amounts never disappear**. Full withdrawals opened on **October 2, 2026** together with the DEX pool; amounts past the 24-hour hold after accrual can be withdrawn on-chain.

Precedence This is the foundation's highest-order policy and takes precedence over any statement on payout or withdrawal elsewhere in this document.

One principle - fixed-quantity payout

WHAT WE PROMISE

7 per day

The count is paid exactly as pre-announced. Price moves never change the count, and total emission is fixed at 2.665B over ten phases.

WHAT WE DO NOT PROMISE

What those 7 are worth

We neither guarantee nor demonstrate the token's value. Price is set by participants trading on exchanges; nowhere in this document is there a target price, return rate or payback period.

ITEM	IF IT WERE VALUE-BASED	FIXED QUANTITY (WLBN)
Payout unit	"so much per day" - count changes daily	"7 per day" - count fixed
If price falls	count rises - emission surge, vicious circle	count unchanged · emission unchanged
If price rises	count falls - no upside for participants	count unchanged · participants benefit
Total emission	Unpredictable	Fixed at 2.665B over ten phases
Nature of the reward	Implies a value guarantee	Consideration for the service of providing data

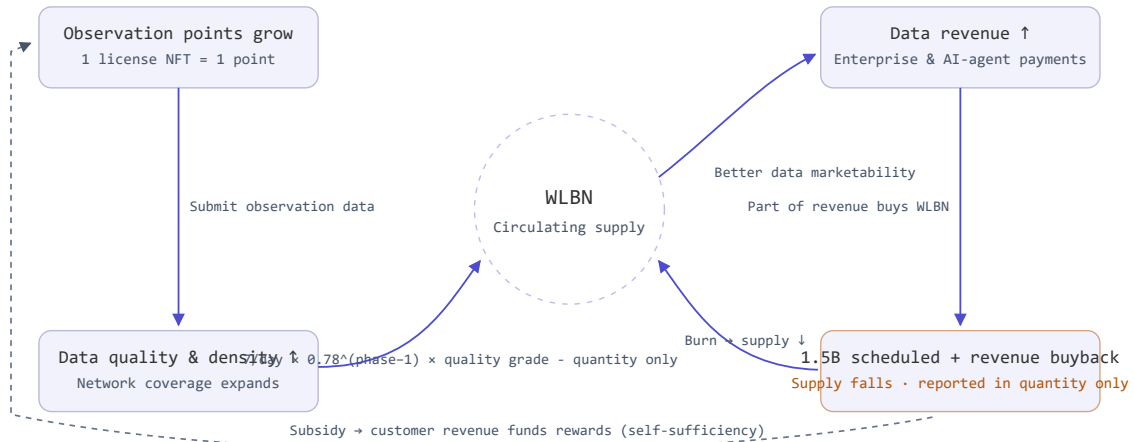


Fig 4. Supply–demand–burn loop · emission decays by phase, burn via buyback and schedule

Allocation of the 10B total

BUCKET	AMOUNT	SHARE	UNLOCK / VESTING
Node-holder rewards, round 1 (Phase 1–5)	1B	10%	Fixed quantities emitted every epoch · Phase 1–5 total ≈ 0.968B
Node-holder rewards, round 2 (Phase 6+)	3.6B	36%	Opens as phase gates are passed after round 1 is spent · unopened amounts convert to burn
Scheduled burn	1.5B	15%	Tranche schedule to be announced separately
Strategic round reserve	1B	10%	Reserved for future strategic rounds · locked until allocated · terms published on allocation
Staking pool	0.8B	8%	Mechanism to be announced
Team & contributors	0.6B	6%	12-month cliff + 48-month linear · full schedule public
Ecosystem & regional partners	0.5B	5%	Conditional unlock per regional-entry milestone

BUCKET	AMOUNT	SHARE	UNLOCK / VESTING
Strategic & private	0.5B	5%	Vesting per investment contract · seed round: 10% at once + 5% monthly in months 7–24 · schedule published
Liquidity & reserve	0.5B	5%	Initial pool deposit and additional deposits
Total	10B	100%	All 10B issued at genesis · no further issuance planned · the issuer account is not blackholed

Daily reward - 22% decay per phase

reward rule

LOG

$\text{daily reward} = \text{base } 7 \times 0.78^{(\text{phase} - 1)} \times \text{quality grade } (0 \sim 1.2)$
 $\text{epoch budget} = \text{active points} \times 7.14 \text{ WLBN} \times \text{phase factor}$

Defined purely in quantity, independent of price. Phases advance not by calendar year but by gates that require both point count and demand conditions, and each phase cuts the per-point quantity by 22%.

PHASE	DAILY BASE	MONTHLY	YEARLY	VS. PHASE 1	PHASE EMISSION
1 Domestic indoor	7.00	213	2,555	100%	64M
3	4.26	130	1,554	61%	187M
5	2.59	79	946	37%	331M
7	1.58	48	575	23%	374M
10	0.75	23	273	11%	273M
Ten- phase total					2.665B

GENESIS ADVANTAGE IS STATED IN QUANTITY ONLY

A genesis holder receives 6.5× the quantity of a Phase-7 participant - not only because of the longer participation, but because the per-phase quantity itself shrinks. What that quantity will be worth later is something we cannot say.

Quality gate - budget fixed, distribution competitive

quality gate

LOG

first pass

normal 7 · excellent 8.4 · failing 0

redistribute

unpaid share pro rata by weight · cap 10.5 per point · excess burned

Failing share	Valid points	Unpaid	Normal, final	Excellent, final
0%	100	0	7.00	8.40
10%	90	71.4	7.78	9.33
20%	80	142.8	8.75	10.50 (cap)
40%	60	285.6	10.50	10.50

Example with 100 points (budget 714 = 100 × 7.14). In every row **total paid + burn = budget**. The worse the quality, the smaller the total emission.

Item	Rule
Payout condition	Transmission rate 95%+ full · 80–95% linear reduction · below 80% nothing
Excellence bonus	+1.4 for the top 10% by data quality - accurate measurement, not clean air
How the grade is derived	Points that pass the transmission gate are scored (uptime, outliers, neighbor correlation, calibration); the top 10% are excellent (1.2), the rest normal (1.0), and gate failures 0
Fraud prevention	Installation photo as the entry gate · ongoing verification via the daily CO ₂ cycle

IMPLEMENTATION STATUS (SEPT 2026)

The current settlement engine accrues 7 per point per day on Phase-1 terms and pays on claim after a 24-hour hold. The excellence bonus (8.4), redistribution of unpaid share and the 0.78 phase factor are confirmed policy and will be rolled into the engine as governance variables. Until then only the first-pass rule (normal 7 · failing 0) applies.

Burn - 15% scheduled + revenue-linked buyback

TYPE	AMOUNT	METHOD
Scheduled burn	1.5B	15% of total supply · tranches announced separately
Buyback burn	Revenue-linked	Bought and burned with 10% of device sales · 10% of access pass sales · 50% of data revenue
Execution	Monthly	Bought quantity · price · transactions published

PRINCIPLE	DETAIL
Revenue-linked	Buyback size is set by revenue ratios - no market intervention to hit a circulation target
All buybacks burned	Buying back to resell is wash trading, not defense
Quality shortfall burned	Amounts above the redistribution cap are burned, not paid
Unopened round-2 burned	Round-2 rewards left unopened at an unpassed gate convert to burn
Published monthly	Executed monthly, with bought quantity, price and transactions published

Burns are reported in quantity only, e.g. "this quarter: 90M emitted · 110M burned". We never say "burning raises the price" - it is an unverifiable claim.

Milestone gates - by achievement, not by year

GATE	SUPPLY CONDITION	DEMAND CONDITION	REWARD WEIGHTING	BUR
G0 Pre-launch	Registration & verification system	1 paying customer	-	-
G1 Genesis opens	Recruitment starts	-	Quality	Non
G2 Liquidity secured	Initial allocation sold	-	Quality	Non
G3 Demand proven	5,000 units	3 B2B deals + repeat purchase	+ scarcity	Non
G4 Sold out	25,000 units · 80% uptime for 3 months	Domestic revenue	Quality + scarcity	Non
G5 First overseas	60,000	Overseas revenue	Same	Non
G6 Outdoor line	120K	Annual revenue ₩1B	+ real usage	Non
G7 Burn begins	220K	Annual revenue ₩2B	All three	Star
G8 Transition	350K	Annual revenue ₩4B	Same	Full s
G9 Self-sufficient	650K	Business revenue funds all rewards	Same	Acco
G10 Mature	1M	Surplus revenue	Same	Max

If the demand condition is unmet, the next gate does not open. The response to a shortfall is to hold the phase and reduce emission.

From subsidy to customer funding

STAGE	POINTS	PHASE EMISSION	FUNDING SOURCE
Phases 1-4	25K → 220K	Rising	Subsidy - funded by hardware business revenue
Phases 5-6	350K → 500K	Peaks, then falls	Transition - data revenue covers a large share
Phase 7+	650K →	Keeps falling	Self-sufficient - customer revenue funds everything · surplus burned

WHY THE CURVES CROSS

Because of the 22% per-phase decay, emission peaks at Phase 6 and then falls, while data revenue keeps growing with point count. From the crossover onward rewards are paid from customer money, not new participation fees. The crossover depends on the data unit price, which is still being validated; it will be recomputed after real Phase-1 contracts and is presented as a goal, not a promise.

Genesis recruitment structure

① PHYSICAL DEVICE

Indoor weather monitor

Measures PM · CO₂ · VOC · temperature/humidity · remains a physical asset

② OBSERVATION-POINT ENTITLEMENT

Private weather-station license

1 NFT = 1 observation point · rights cannot be sold separately

③ DATA-PROVISION REWARD

7 WLBN per day

Quality grade applied · 22% decay per phase

③ PURCHASE PRIORITY

Priority purchase right for follow-on lines

Outdoor line offered first · a queue position, not a profit right

SALES ROUND	TIMING	PRICE	PURCHASE LIMIT
Round 1 Genesis sale	2026-09-15 (reservations Sep 7-14)	450 RLUSD · KRW 750,000 (list price 650 RLUSD)	10 per account
Round 2 general sale	From 2026-10-03 09:00 KST until stock runs out	650 RLUSD · KRW 1,083,000	10 per individual · 30% off for dealers buying 100+ units cumulatively

The purchase limit is 10 units per individual, and dealers buying 100 units or more cumulatively receive a 30% discount. A fixed daily reward per NFT is paid while a device NFT has been issued but the device is not yet installed; once installed, operation and quality criteria apply.

Operating principles and disclosure policy

LIQUIDITY DEPLOYMENT PRIORITY	DURATION	WHO BENEFITS
① Market depth	Permanent	Everyone who trades
② Buyback burn	Permanent	All remaining holders
③ Bid buffer	Temporary	Whoever sells at that moment

10% of device sales, 10% of AI Factory access pass sales and 50% of data revenue buy back and burn WLBN (executed monthly, records published). Payment structure: enterprise customers pay in KRW or RLUSD, AI agents pay automatically, and the protocol buys WLBN with part of revenue. Customers are never asked to buy the token to pay, and allocation ratios are announced in advance.

ITEM	PUBLIC	NOT PUBLIC
Liquidity	Pool balance on-chain at all times · deployment rules · burn record	Bid prices · buyback timing
Vesting	Full schedule · unlock amounts and dates	-
Reward	Quantities paid · quality criteria · redistribution results	-

A published bid is a free put option to the market, steering funds to short-term traders rather than long-term holders. **Five things we do not do:** use node count as a KPI · open the next phase without demand · list on centralized exchanges before demand is proven · pay every dataset the same · make the subsidy permanent.

The metrics we watch are sell-through, uptime and data revenue - not the token price. If sales miss the target we cut emission rather than defend the price.

Operations

Emission Simulation
- Phase-Decay Model

Ten-phase cumulative emission and the self-sufficiency crossover are computed with a phase-decay model. Every figure derives from the published policy parameters (base 7 · decay 0.78 · points per gate) and is neither a forecast nor a promise of reward level or income. Actual receipts depend on quality grade and redistribution.

Conclusions first

- FINDING 01

Emission peaks at Phase 6

Point count keeps growing, but per-point quantity falls 22% per phase, so phase emission peaks around Phase 6 (near 374M) and then declines, while data revenue keeps rising with point count.
- FINDING 02

Genesis = 6.5× a Phase-7 entrant

Cumulative receipt is 10,646 for a Phase-1 participant versus 1,647 for a Phase-7 entrant. The gap comes more from shrinking per-phase quantity than from longer participation.

FINDING 03

Worse quality, less emission

The epoch budget is fixed, but redistribution is capped at 10.5 per point, so with 40% failing points 12% of the budget is burned. **Paid + burned = budget.**

FINDING 04

No demand, no emission

Advancing a phase requires both point count and demand. On a demand shortfall the phase is held and emission cut, structurally preventing supply from running ahead of nodes.

Model structure

core model equations

LOG

```
reward(p, g)   = 7 × 0.78^(p - 1) × g           g ∈ { 0, 1.0, 1.2 }
budget(p, n)   = n_active × 7.14 × 0.78^(p - 1)   fixed per epoch
redistribute    = unpaid × w_i / Σw               cap 10.5 per point · excess →
emission(p)    = Σ_epochs budget(p, n) - burn(p)
Σ p=1..10      = 2.665B ≤ 4.6B (node rewards: round 1 1B + round 2 3.6B)
```

Confirmed parameters

ITEM	VALUE	RATIONALE
Base reward	7 per day (normal grade)	Pre-announced quantity · independent of price
Phase decay	0.78 (22%)	Keeps the ten-phase total within the 4.6B round-1 + round-2 node budget
Quality grade	0 · 1.0 · 1.2	Failing · normal · excellent (top 10%, +1.4)
Redistribution cap	10.5 per point	Prevents concentration · excess burned

ITEM	VALUE	RATIONALE
Phase-1 points	25,000	Korea indoor · round-1 Genesis sale + round-2 general sale
Ten-phase cap	2.665B	Round 1 1B (Phase 1–5) + round 2 (Phase 6–10)
Genesis bonus	+20%	Genesis membership holders receive an extra 20% on the rewards of the device NFTs they hold
Round-2 rewards	3.6B	Opens as phase gates are passed after round 1 is spent · unopened amounts burned

STATED ASSUMPTIONS

The self-sufficiency point depends on the data unit price, which is still being validated and will be recomputed after real Phase-1 contracts. Table values follow from the equations above with the per-gate point counts and match the published deck. They are goals, not promises.

Operations

Residual Risks

Items to recognize and manage during implementation.

ITEM	RISK	MITIGATION	PRIORITY
Device identity	Verifying NFT ownership alone lets a stolen seed inject forged data without limit	Payload signature verification + time-series anomaly detection + neighbor-node cross-validation	Recommended
Oracle trust	The off-chain API is a single trust point - data does not live on the ledger	Record per-batch Merkle roots in Memos to preserve after-the-fact auditability	Recommended
Reserve exhaustion	Accumulated TrustLines and Offers starve the Hot	Balance threshold monitoring + a batch	Recommended

ITEM	RISK	MITIGATION	PRIORITY
	wallet of XRP → every transaction fails	job to clear unfilled Offers	
Price volatility	The quantity received is fixed, but its value rises and falls; downside risk rests entirely with the holder	Fixed-quantity payout · no value guarantee · no target price or return	Acknowledged
Business shortfall	If data does not sell, the next phase does not open and reward quantities shrink	Demand conditions on every gate · emission cut on shortfall	Mandatory
Operation obligation	Installed points that stop operating earn nothing (NFTs awaiting installation receive the fixed daily amount); transmission below 80% also pays nothing	Graded payouts · pre-announced criteria · installation photo and daily CO ₂ -cycle verification	Mandatory
Early trading conditions	Right after launch trading is thin, so small trades can move the price sharply	Initial pool deposit, then additional deposits · large trades separated · centralized exchange listing only after the gates	Recommended
Indoor data sensitivity	CO ₂ and PM time series reveal occupancy count, sleep times, and absence periods	5-stage de-identification + k-anonymity + differential attack defenses. Privacy legal review must come first	Mandatory
Regulatory change	Crypto-asset regulation differs by jurisdiction and changes	Local rules are reviewed region by region to decide whether and how to enter · utility nature	Mandatory

ITEM	RISK	MITIGATION	PRIORITY
		maintained (no equity or dividends)	
Hot wallet key management	A leaked Hot Wallet seed drains the reward pool balance immediately	KMS-backed signing separation · keep a balance ceiling · never load the Issuer (cold) seed at runtime	Mandatory
RLUSD issuer dependency	A freeze or policy change by the RLUSD issuer (rMxCK...) can block the payment and settlement path	Monitor TrustLine state + predefine an alternate settlement-currency path	Recommended
Outbox double payment	Resubmitting unvalidated transactions after a cron restart risks double payment	Outbox idempotency keys + pre-submit tx lookup + reconcile cron cross-check	Mandatory
Failure during the claim hold	A settlement failure inside the 24-hour hold leaves accruals stuck as unclaimable	The reconcile cron cross-checks the accrual ledger against on-chain results and auto-recovers unpaid amounts	Mandatory
Sensor calibration drift	Low-cost MOX and light-scattering sensors accumulate deviation over months, degrading data value	The <code>calibration</code> term (15 points) in the quality score + automatic correction factors from neighbor-node correlation	Recommended
x402 request header	This implementation uses the custom <code>X-Payment-Tx: <hash></code> header, alongside a prepaid-credit path (<code>X-Api-Key</code> · <code>/api/data/topup</code>)	Standard x402 uses the <code>X-PAYMENT</code> header. The response body already follows the standard schema, so swapping the header completes alignment	Acknowledged

Operations

Execution Checklist

Each Step proceeds to the next only after approval. Completion criteria are stated alongside.

Step 1 - XRPL Foundation & Wallets

- ☐ Mainnet WebSocket (`wss://xrplcluster.com`) client singleton + reconnection handling
Done when: `server_info` response received
- ☐ Prepare three wallets - Issuer (cold, rDjz8...) / Hot (rhTTt...) / Treasury (rPZFY...) - and fund the XRP reserve
- ☐ `.env` + `.env.example` set up, listed in `.gitignore`, validated with zod
- ☐ Shared `submit()` wrapper enforcing `tesSUCCESS` + `validated`

Step 2 - Wellbian Token

- ☐ Currency hex constants fixed (WLBN / RLUSD)
- ☐ Issuer `AccountSet` - `asfDefaultRipple` **enabled**
- ☐ Hot-Treasury-user-wallet TrustSet - `tfSetNoRipple` applied
- ☐ Confirm all 10B WLBN issued (no further issuance)
Done when: balance confirmed via `account_lines`

Step 3 - License NFT

- ☐ IPFS metadata schema fixed and uploaded (production URI pinned)
- ☐ Full flow: `mintLicense` → `offerLicenseTo` → `acceptLicense`
- ☐ `verifyDeviceLicense()` - checks both Issuer and Taxon
- ☐ Negative test: a non-holding address resolves to `valid:false`

Step 4 - Oracle & Rewards

- ☐ `GET /v1/c` ingest proxy implemented + 5-stage receipt validation

- ☐ Order respected: signature check → license lookup (cheapest check first)

- ☐ 403 on license verification failure, with no reward paid

- ☐ Absolute humidity and dew point derived before storage
Done when: `absoluteHumidity(20,60)≈10.35`, `(30,60)≈18.21`

- ☐ Daily settlement - paid + burned = epoch budget (active points × 7.14 × phase factor), no point above 10.5
Done when: paid + burned equals the budget for failing-point shares from 0% to 40%

- ☐ Claim API - only accruals past the 24-hour hold, paid as a single aggregated WLBN Payment

- ☐ Reconcile cron - cross-checks on-chain results against the accrual and payout ledger, auto-recovering mismatches

- ☐ Batch Merkle root recorded in `Memos`, Merkle proof verified

- ☐ Hot wallet outbox pattern applied · resume-after-stop verified
Done when: zero double payments and zero Sequence errors on resume

Step 5 - AMM

- ☐ `AMMCreate` WLBN/RLUSD mainnet pool created (TradingFee 0.5%, swap-only)

- ☐ `AMMDeposit` additional liquidity supplied and LPToken receipt confirmed

- ☐ RLUSD→WLBN swap via `path_find` + `SendMax`

- ☐ Pool ratio change recorded via `amm_info` before and after the swap

Step 6 - x402 Gateway

- ☐ Request without header → 402 + `accepts` array payment terms

- ☐ `verifyPayment()` - validates recipient, issuer, currency, delivered amount, tesSUCCESS

- ☐ Judgement based on `delivered_amount` (PartialPayment defense)

- ☐ Nonce store blocking tx_hash reuse

- ☐ Agent E2E (real mainnet payment): 402 → payment → retry → 200
Done when: one script run completes the whole flow unattended

- ☐ Burn batch job (scheduled · buyback · quality shortfall), publication of burn quantity and tx, circulating-supply reduction verified

IAQ Spec - Data Layer

- ☐ IAQ telemetry schema v1 fixed + device signature format

- ☐ Taxon separation (`IAQ 3026` / `OAQ 2026`), quality grades (0 · 1.0 · 1.2) applied

- ☐ `qualityScore()` computes all four terms - especially neighbor-node correlation

- ☐ De-identification pipeline P1-P5, epoch salt rotation verified

- ☐ k-anonymity rollup - refuse the response if geohash3 is still short
Done when: a query on a 4-node cell either rolls up or is refused

- ☐ All four differential attack defenses (rounding · filter cap · query budget · minimum cohort)

- ☐ Schema-level guarantee that cohort responses **contain no node list**

- ☐ Prepaid credit - top-up, signature-based deduction, monotonic nonce
Done when: replaying the same nonce is rejected with `BAD_NONCE`

- ☐ Redistribution of unpaid share - 10.5-per-point cap, excess-burn verified

OPERATIONAL VERIFICATION SCENARIO

1. Send IAQ telemetry from an unauthorized address → **403** (no license)
2. Issue the NFT and resend from the same address → **202 Accepted**, derived absolute humidity confirmed
3. Run the epoch batch → per-node quality scores and share allocation printed, **total = budget** confirmed
4. Duplicate the node count 10× and re-run → **budget = active points × 7.14 × phase factor** scales proportionally, per-point quantity unchanged

5. Manufacturer submits a cohort query → **402**, pays, retries → **200 + size only**
6. Narrow the filters and re-query → **422 COHORT_TOO_NARROW** (actual size not disclosed)
7. Call again with the same tx_hash → **402 (HASH_ALREADY_USED)**
8. Burn batch → **circulating supply falls** via Issuer TrustLine offset

Operations

Glossary

TERM	DEFINITION
DePIN	Decentralized Physical Infrastructure Network. A model that decentralizes the buildout and operation of physical infrastructure (sensors, base stations) through token incentives.
XLS-20	XRPL's native NFT standard. Provides dedicated transaction types such as <code>NFTokenMint</code> , requiring no smart contract.
XLS-30	XRPL's native AMM standard. Order book and AMM liquidity are automatically routed together.
Issued Asset (IOU)	A token in the form of a claim against the issuer account. Created when the issuer sends a Payment - there is no separate mint transaction.
TrustLine	An on-chain relationship declaring how much of a given issuer's currency a holder will accept. Without it you get <code>tecNO_LINE</code> .
Rippling	XRPL's distinctive mechanism linking TrustLines of the same currency so balances move automatically. It is the basis for issuer-routed transfers and also the abuse vector.
x402	An M2M payment protocol that puts HTTP 402 Payment Required to actual use. The server states payment terms and the client retries with on-chain proof of payment.
delivered_amount	The actual delivered amount recorded in transaction metadata. Verification uses only this value, to block PartialPayment bypass.

TERM	DEFINITION
Destination Tag	A 32-bit integer identifier attached to a Payment. Distinguishes purpose and customer among payments arriving at the same address.
IAQ	Indoor Weather (indoor weather). In this document, the collective term for the six metrics PM2.5 · PM10 · CO ₂ · TVOC · temperature · humidity.
Absolute humidity (AH)	Actual water vapor mass per unit volume (g/m ³). Unlike temperature-dependent relative humidity it expresses dehumidification load directly, making it the input for dehumidification algorithms and condensation prediction.
TVOC	Total volatile organic compounds. MOX sensors output a relative index, so it is used for trend and spike detection rather than read as an absolute concentration.
geohash	A grid scheme encoding latitude/longitude as a string. Fewer characters mean a larger area (5 chars ≈ 4.9 km, 3 chars ≈ 156 km).
k-anonymity	The property that no combination of attributes narrows a set below k members. This specification sets k=5 as the floor and 25 after rounding as the practical floor.
Differential attack	An attack that isolates individual subjects from the differences between repeated queries with slightly varied filters. Defended with count rounding, a filter count cap, and query budgets.
Epoch	The reward settlement period - a fixed one-day unit bounded by KST midnight, on which the epoch budget (active points × 7.14 × phase factor) is paid and redistributed by quality grade.
Prepaid credit	Topping up a balance with a single on-chain payment, then deducting from it by signature alone on subsequent calls. Avoids ledger finality latency on high-frequency APIs.
RLUSD	The settlement currency of this ecosystem. Its issuer is <code>rMxCKbEDwqr76QuheSUMdEGf4B9xJ8m5De</code> , which is not the WLBN Issuer.
Issuer · Hot · Treasury wallets	Issuer (cold, <code>rDJz8WJhsKgydqJzSZXMpsJot3eRmSkR5</code>) is issuance-only and never loaded at runtime; Hot (<code>rhTTtx8YyKzcmPDsoroYXuHdGMThrnweb1</code>) handles payout and receipt; Treasury (<code>rPZFY2yLgqjmBrxtxwHXnB1doBt7gu7xin</code>) holds reserves.

TERM	DEFINITION
Claim hold	The 24-hour waiting period before epoch-settled accruals become claimable. Accruals past the hold are paid as a single aggregated WLBN Payment.
Outbox pattern	Hot wallet transactions are recorded to an outbox first and drained sequentially by a single worker. If the cron stops, it resumes from incomplete entries, preventing double payment.
Taxon scheme	Observatory 1001·1002·1003; stations IAQ 3026 · OAQ 2026 · AI infra 4026 · radar 5026 · satellite 6026; voucher 1010.
Supported wallets	Embedded wallet (ID + password / Google), D'CENT mobile in-app, Xaman, GemWallet, Crossmark.
Phase · phase factor	An emission stage entered through a gate requiring both point count and demand. The phase factor $0.78^{(\text{phase}-1)}$ multiplies the per-point quantity, a 22% decay per phase.
Quality grade	A 0 · 1.0 · 1.2 factor set by the transmission gate and quality score: failing 0, normal 1.0 (7/day), top-10% excellent 1.2 (8.4/day).
Redistribution	Splitting the unpaid remainder of the epoch budget pro rata by weight after the first pass, capped at 10.5 per point with the excess burned so that paid + burned = budget.
Milestone gate	Phase-entry conditions G0–G10 pairing supply (points, uptime) with demand (paying customers, revenue); on a demand shortfall the phase is held and emission cut.
Buyback burn	Burning WLBN bought on the market with a fixed share of revenue by returning it to the Issuer. All buybacks are burned, with quantity, price and transactions published monthly.
Revenue-linked burn	The operating rule of buying and burning WLBN with 10% of device sales, 10% of access pass sales and 50% of data revenue; executed monthly with records published.
Blackhole account	An account made permanently unable to sign by setting its Regular Key to an invalid address and disabling the master key. This system's Issuer is not a blackhole; burning works by offsetting on return to the Issuer.

Indoor & Outdoor Weather Data Ecosystem (IAQ / OAQ)

An ecosystem that collects indoor (IAQ) and outdoor (OAQ) weather data, supplies it to appliance manufacturers for R&D and marketing and to environmental services, and returns on-chain rewards to contributors.

XRP Ledger mainnet (live since 2026-08-13) · XLS-20 NFT · Issued Asset · XLS-30 AMM · x402

paper.wellbianlabs.io · wellbian.io/docs/device-api.html · wellbian.io/data
· wellbian.io/redeem

The reward figures in this document are planning baselines. They guarantee neither returns nor principal, and present no fiat-equivalent value for Wellbian (WLBN).