Urd Atlas — Methodology & Data Reference Whitepaper

Methodology & Data Reference Whitepaper

Document version: 1.1 Methodology version: v1.2 Publication date: May 2026 Coverage start: 01 December 2024 for all supported chains Supported chains: Bitcoin, Ethereum, Arbitrum, Base Publisher: Urd Atlas · urdatlas.com Contact: support@urdatlas.com Commercial/legal entity: Disclosed in commercial terms, order flow, and invoice documentation at time of purchase. This whitepaper is not the commercial contract. Classification: Public — Reference distribution Product boundary: No price data. No forecasts. No recommendations.

This document is the public methodology and data-reference whitepaper for the Urd Atlas data product. It is intended for technical due diligence, compliance review, research workflow evaluation, and product integration. It describes what Urd Atlas publishes, what the fields mean, how regime labels should be interpreted, what is verifiable from public artifacts, and which analytical boundaries apply.

Document version and methodology version are tracked separately. This document version updates and clarifies the public product description, API delivery model, Regime Briefs layer, sample-pack role, and operational trust controls. It does not change the classification methodology, thresholds, confidence gate, field semantics, or historical row interpretation governed by methodology version v1.2.

Non-advisory notice

Urd Atlas publishes descriptive blockchain regime reference data. It does not publish price data, price forecasts, return forecasts, trading signals, portfolio recommendations, investment advice, or any instruction to buy, sell, hold, avoid, rebalance, hedge, or allocate capital.

All outputs are chain-relative descriptions of observed network conditions under the published methodology. Users are responsible for their own downstream interpretation, filtering, integration, compliance controls, and decision frameworks.

Abstract

Urd Atlas publishes daily blockchain regime reference data for Bitcoin, Ethereum, Arbitrum, and Base. The product is designed for users who need a stable, auditable, machine-readable state variable describing observed chain conditions, rather than a narrative dashboard or trading signal.

Each publication cycle produces three canonical machine-readable data layers per supported chain:

1. Gold — direct daily on-chain observations and robust daily summaries.
2. Derived — deterministic rolling transforms of Gold, primarily short- and medium-window context fields.
3. Meta — analytical classification layer containing regime label, confidence, scorecard, drivers, freshness fields, methodology version, and integrity fields.

In addition, Urd Atlas publishes Regime Briefs: derived explanatory summaries built from the same evidence surface used by the canonical layers. Briefs are designed for readability and workflow context. They do not replace Meta, do not introduce advisory interpretation, and do not alter the canonical data contract.

The Meta layer exposes a categorical regime label drawn from a five-state vocabulary: STABLE, HEATING, CONGESTED, CHEAP, and UNKNOWN/DEGRADED. All Meta rows are identified at minimum by `chain`, `date`, and `methodology_version`. Named-state rows add `regime.determinism_hash`, which anchors the public named-state regime payload. UNKNOWN/DEGRADED

rows do not carry a determinism hash because no named-state payload is being asserted; they are instead identified and compared using their public degradation fields, including `updated_through`, `confidence.confidence_score`, and `status.label`.

The product is descriptive and chain-relative. It is intended for integration into external research, data-engineering, monitoring, and compliance workflows where users require explicit provenance, versioning, confidence, and evidence fields.

Keywords: blockchain analytics, regime classification, on-chain data, reference data, confidence gate, deterministic publication, audit trail, JSON API, provenance, Bitcoin, Ethereum, Arbitrum, Base.

Table of Contents

Reading path

Technical buyers: Prioritize Sections 4, 7, 9, 11, 13, 15, and 16. Compliance and audit reviewers: Prioritize Sections 2, 13–19, and 21. Research users: Prioritize Sections 9–12, 15, and Appendix A. API integrators: Prioritize Sections 4, 16, 20, and Appendix B.

1. Executive Summary

Urd Atlas is a deterministic blockchain regime reference data service. Its purpose is narrow: to classify observed on-chain network conditions into a small number of descriptive states that can be joined into a customer's own analytical system.

The product does not attempt to predict asset prices or user behavior. It does not rank assets. It does not provide recommendations. It answers a more constrained question:

> *Is this blockchain currently behaving within its recent normal operating range, or has the observed on-chain condition shifted into a materially different regime under the published methodology?*

Urd Atlas is built for users who want:

- a daily chain-state variable,
- JSON artifacts that can be joined on `chain` and `date`,
- explicit confidence fields,
- evidence drivers,
- freshness metadata,
- method/version traceability,
- a public sample pack for diligence,
- and stable boundaries between public methodology, customer data delivery, and proprietary implementation.

The commercial deliverable is not a dashboard narrative. The commercial deliverable is the published data contract: Gold, Derived, Meta, and supporting Briefs. The dashboard and documentation exist to make that data contract understandable and inspectable.

2. Product Mandate and Analytical Boundary

Urd Atlas classifies observed blockchain network state. It does not infer future asset performance.

The product's mandate is to publish daily, chain-relative regime reference data under a documented methodology. A named regime label means that the observed evidence surface satisfied the published rules and confidence gate for that chain and date. It does not mean that an asset is expected to rise, fall, outperform, underperform, or produce any particular return profile.

Mandate statement

Urd Atlas classifies observed blockchain network state. It does not infer future asset performance. It does not rank assets. It does not tell a user what to buy, sell, hold, short, hedge, or avoid.

Intended users

The primary user is a research desk, quantitative analyst, data engineer, compliance reviewer, institutional diligence team, or advanced independent analyst. The product is especially relevant when the user needs a versioned, auditable, daily state variable that can be joined into an existing workflow.

Design commitments

Commitment	Operational meaning
Strict descriptiveness	Labels describe observed chain conditions relative to the chain's own recent history. No cross-chain absolute ranking is implied.
No price dependency	Published artifacts do not include price observations, return forecasts, or price-derived labels.
Epistemic conservatism	When evidence is insufficient, the system publishes UNKNOWN/DEGRADED rather than forcing a named label.
Deterministic publication	Named-state rows carry a deterministic hash over the public regime payload.
Methodology transparency	Field semantics, thresholds, label order, confidence gate, and interpretation boundaries are documented publicly.
Versioned contract	Field meaning and analytical semantics are governed by `methodology_version`.
Archived-as-published principle	Historical artifacts are retained as-published unless a correction is explicitly issued and documented.
Machine-readable first	The JSON artifact is the product; the website and Briefs are explanatory surfaces around it.

3. Supported Chains and Publication Cadence

Urd Atlas currently supports four chains:

Chain	Public identifier	Typical freshness expectation
Bitcoin	`bitcoin`	Approximately one day
Ethereum	`ethereum`	Approximately one day
Arbitrum	`arbitrum`	Approximately seven days
Base	`base`	Approximately seven days

The cadence is daily, not intraday. The methodology is intentionally designed around daily observations and recent historical context. Intraday spikes are not intended to flip regimes in real time. Users requiring intraday monitoring should treat Urd Atlas as a daily reference layer, not as a tick-level alert system.

Historical coverage

The public methodology archive begins on 01 December 2024 for all currently supported chains. Coverage depth may expand over time. Customer access to historical files can depend on product tier and commercial terms, but the methodology coverage start defines the earliest intended public archive date for the current supported-chain set.

L2 freshness clarification

The longer expected lag for Arbitrum and Base is a product-publication lag, not a statement about L2 finality or the native chain's ability to produce near-real-time data. It reflects the current data-source availability, ingestion cadence, validation window, and operational publication model used by Urd Atlas. A lag near the expected value for ARB or BASE is normal under the current service model and is not, by itself, a pipeline failure.

4. Product Architecture

Each publication cycle produces three canonical machine-readable artifact layers per chain:

Layer	Role	Interpretation rule
Gold	Daily observation layer	Direct daily chain aggregates or robust daily summaries. No regime logic, confidence degradation, or categorical interpretation.
Derived	Deterministic transform layer	Rolling transforms built from Gold, primarily MA7 and MA30 features used for trend context.
Meta	Analytical classification layer	Regime label, confidence, scorecard, driver set, freshness fields, methodology version, and integrity fields.

A fourth layer, Regime Briefs, is published for explanatory use:

Layer	Role	Interpretation rule
Briefs	Explanatory publication layer	Human-readable summaries derived from the published evidence surface. Briefs do not replace Meta and do not introduce advisory interpretation.

Separation of concerns

Gold answers: what was observed? Derived answers: how does the observation compare to recent time context? Meta answers: which regime label is supported under the published methodology? Briefs answer: how can the published state be summarized for a reader without changing its meaning?

This separation is central to the product. Raw observations, deterministic transforms, analytical classification, and explanatory summaries can be inspected independently.

Canonical join keys

The intended integration model is straightforward:

```
chain + date
```

Users can join Gold, Derived, Meta, and downstream internal data on those keys. The `methodology_version`, `ruleset_id`, `updated_through`, and integrity fields should be preserved in downstream storage.

5. Gold Layer

Gold publishes direct daily chain observations or robust daily summaries for each chain. Gold does not apply regime logic, confidence gating, smoothing, or categorical interpretation. It is the measurement layer used by Derived and Meta.

Gold field families

Field family	Public meaning	Verification class
Daily counts	Daily transaction volume and block production activity, including `tx_count_daily` and `block_count_daily`.	B
Native value and fee fields	Native-denominated same-day transfer and fee measures, including `median_tx_value_native` and `median_tx_fee_native`.	B
Execution-quality and capacity fields	Daily failure burden or capacity usage where semantically meaningful, including `failed_tx_rate` and `gas_utilization_pct`.	B
Breadth and cadence fields	Participation breadth and typical inter-block interval, including `unique_active_addresses` and `avg_block_time_sec`.	B

Global interpretation rules

- All dates are UTC calendar dates.
- All interpretation is chain-relative, not cross-chain absolute.
- No price conversion is applied inside published artifacts.
- Unsupported or unreliable fields are published as `null`.
- If field meaning changes materially, `methodology_version` must change.

6. Derived Layer

Derived fields are deterministic transforms of Gold. The core public pattern is the rolling average family:

```
<metric>__ma7
<metric>__ma30
```

These fields provide short- and medium-term context for charting, trend interpretation, and downstream workflows.

Transform	Definition	Interpretation
`__ma7`	7-day simple moving average of the corresponding Gold metric.	Short-term smoothed condition.
`__ma30`	30-day simple moving average of the corresponding Gold metric.	Medium-term smoothed condition.
Archive-start handling	Available observations are used near archive start rather than forcing null solely due to insufficient lookback depth.	Preserves continuity at archive start.

Derived is published separately from Gold because smoothed series are derivative quantities. Separating them makes derivation explicit and prevents raw daily observations from being confused with trend-smoothed context values.

7. Meta Layer

Meta is the primary analytical layer. It translates the observation and transform layers into a named state through explicit evidence fields rather than opaque natural-language interpretation.

A Meta row is designed to be joined into a user's downstream dataset and filtered by confidence, chain, ruleset, freshness, and methodology version.

Component	Representative fields	Purpose
Regime state	`status.label`, `regime.label`, `status.one_liner`	Top-line state and compact explanation.
Confidence	`confidence.confidence_score`, `data_quality_score`, `label_confidence_score`	Evidence strength and publication eligibility.
Publication gate	`publish_confidence.threshold`	Named-state gate value.
Scorecard	`scorecard.dimensions`	Demand / Friction / Capacity display scores.
Drivers	`regime.drivers[]`	Ranked evidence variables with z-score, percentile, momentum, and current value.
Freshness	`updated_through`, `lag_days_vs_utc_today`	Timeliness of supporting evidence.
Methodology	`methodology_version`, `ruleset_id`, `window_days`	Versioned interpretation contract.
Integrity	`regime.determinism_hash`	Named-state payload anchor. Present only on named-state rows.

Meta is the only canonical interpretive layer. Its interpretation is constrained by rules, thresholds, confidence gates, and documented boundaries.

8. Regime Briefs

Regime Briefs are derived explanatory publications built from the same evidence surface as the canonical data layers. They are designed for readability, product navigation, and analyst workflow support.

Briefs do not replace Gold, Derived, or Meta. They do not create a separate advisory channel. They do not change the meaning of the regime label. They are summaries of the already-published evidence.

Briefs role

Briefs function	Meaning
Readability	Converts structured evidence into concise human-readable context.
Navigation	Helps users identify which chain/date/state deserves closer inspection.
Diligence support	Gives a first-pass explanation before a user inspects Meta fields directly.
Non-advisory boundary	Briefs must remain descriptive and must not recommend user action.

Briefs interpretation rule

When a Brief and a Meta artifact refer to the same chain/date, Meta is the canonical machine-readable source. If there is any ambiguity, users should interpret the Brief as an explanatory surface and the Meta row as the governing data artifact.

9. Confidence System

The confidence system is the primary epistemic control in Urd Atlas. It serves two functions:

1. It gates publication eligibility, preventing named labels when the evidence surface is too weak.
2. It degrades displayed scorecard scores toward neutral when confidence is reduced but remains above the gate threshold.

Composite confidence score

The published confidence score is the geometric mean of two component scores:

Component	Symbol	Definition
Data quality score	Q	Measures data sufficiency: completeness, freshness, density of recent history, and coverage of required metrics. Range: [0, 1].
Label confidence score	L	Measures analytical clarity: how strongly the current evidence distinguishes the published label from alternatives. Range: [0, 1].
Composite confidence score	C	`C = sqrt(Q × L)`. Both dimensions must be adequate for the composite to be high. Range: [0, 1].

The geometric mean is intentionally conservative. If either component is zero, the composite confidence is zero. If one component is low, the other component can raise the composite only partially; it cannot erase the low-confidence dimension. This is intentional: strong label clarity should not fully compensate for poor data quality, and strong data quality should not fully compensate for weak label clarity.

Publication gate

The canonical publication gate threshold is:

```
0.40
```

When the composite confidence score falls below this threshold, the product publishes UNKNOWN/DEGRADED as the regime label regardless of what the underlying axis scores indicate.

Confidence bands

Band	Score range	Interpretation
Good	`>= 0.70`	Label is well-supported by the current evidence surface.
Caution	`0.40 – 0.69`	Label is published, but scorecard scores are pulled toward neutral. Cross-check before strong interpretation.
Degraded	`< 0.40`	Gate blocks named label. UNKNOWN/DEGRADED is published.

Scorecard confidence degradation

```
published_score = 50 + (score_raw − 50) × effective_confidence
```

At confidence 1.0, the published score equals the raw score. At confidence 0.0, the published score equals 50. Intermediate values produce proportionally attenuated scores.

This formula is Class A verifiable when the relevant fields are present in the published artifact.

10. Scorecard Architecture

The scorecard compresses current chain conditions into three analytical axes:

1. Demand
2. Friction
3. Capacity

Scores are chain-relative and bounded to a 0–100 display scale, with 50 as neutral for that chain's recent history.

The scorecard is not a separate label engine. It is an evidence surface that the regime classification engine reads alongside driver evidence.

Axes

Axis	Primary signal	Key metrics
Demand	Transaction throughput and network utilization pressure.	`tx_count_daily`, `unique_active_addresses`
Friction	Cost burden relative to value transferred.	`median_tx_fee_native`, `fee_burden_proxy`
Capacity	Capacity utilization and block production regularity.	`gas_utilization_pct` for EVM chains; `blocktime_instability` proxy for Bitcoin

Normalization and score construction

Step	Operation	Formula / parameter
1	Input smoothing	7-day rolling average of the raw Gold metric.
2	Baseline	365-day baseline excluding the most recent 14 days.
3	Robust z-score	`(x − median) / (1.4826 × MAD)`
4	Bounded display score	`score_raw = 50 + 40 × tanh(z / 1.5)`
5	Confidence degradation	Pull published score toward neutral by confidence.

Scorecard z-score and driver z-score are distinct

`regime.drivers[].z_robust` and scorecard dimension scores are not the same calculation. They use different input series, windows, and analytical purposes. They should not be expected to match for the same metric on the same date.

Bitcoin capacity proxy

Bitcoin does not expose EVM-style gas utilization. Bitcoin Capacity therefore relies on block-time instability.

```
blocktime_instability =
rolling_mean(
```

```
    |avg_block_time_sec - rolling_median_30(avg_block_time_sec)|
    / rolling_median_30(avg_block_time_sec)
  )
```

A high BTC Capacity score means block timing is unusually erratic relative to recent norms. It does not necessarily mean blocks are directionally slow.

11. Regime Classification Engine

The classification engine maps the current evidence surface to a mutually exclusive label. The vocabulary is intentionally small because the product is designed to publish stable reference states rather than a dense taxonomy of narrative conditions.

Five-state label vocabulary

Label	Plain meaning	Trigger condition summary
STABLE	No axis is in a notably elevated or depressed state.	None of CONGESTED, CHEAP, or HEATING conditions hold.
HEATING	Demand is elevated and directional trend confirms acceleration.	Demand axis HIGH and at least one axis shows positive momentum.
CONGESTED	Network is under significant pressure.	Capacity EXTREME_HIGH, or Capacity HIGH and Friction HIGH.
CHEAP	Friction and capacity pressure are both low.	Friction LOW and Capacity LOW.
UNKNOWN/DEGRADED	Evidence quality is insufficient for a named label.	Confidence score below gate threshold.

Threshold band assignment

The classification engine assigns each axis to a band using two parallel criteria:

1. 90-day percentile rank
2. robust z-score

A band fires when either criterion is met.

Under methodology version v1.2, the public band cutoffs below are part of the published methodology contract. Chain-specific `ruleset_id` values can define which metrics, proxies, and axes are relevant for a chain, but the public band thresholds should be read as v1.2 methodology-level thresholds unless a future methodology version or explicit ruleset document states otherwise.

Band	Percentile	Z-score	Role
EXTREME_HIGH	`>= 95th`	`>= 2.5`	Can trigger CONGESTED alone on Capacity.
HIGH	`>= 80th`	`>= 1.5`	Feeds CONGESTED and HEATING.
NORMAL	`20–80th`	`-1.5 to +1.5`	Default, no band condition.
LOW	`<= 20th`	`<= -1.5`	Feeds CHEAP.
EXTREME_LOW	`<= 5th`	`<= -2.5`	Stronger low signal; also feeds CHEAP.

Label evaluation order

Labels are evaluated in a fixed priority order:

1. UNKNOWN/DEGRADED
2. CONGESTED
3. CHEAP
4. HEATING
5. STABLE

The first item in this list is the confidence gate described in Section 9, expressed as a publication state. If the composite confidence score is below 0.40, UNKNOWN/DEGRADED is published and no named-label rules are evaluated for publication. If the row passes the gate, the named-label checks then proceed in the order shown above. Once a condition is satisfied, subsequent checks are skipped.

Label stability and persistence

Urd Atlas does not apply a universal fixed multi-day confirmation window across all labels.

- HEATING requires directional trend confirmation and will not appear on a single-day elevation alone.
- CONGESTED and CHEAP do not require trend confirmation and can fire on a single-day threshold crossing.
- STABLE fires as the residual default.

Downstream consumers using labels as period classifiers should apply their own minimum-duration filter if multi-day stability is required.

12. Freshness Policy

Freshness and confidence are related but separate. Freshness describes how recently a row has been published relative to expected cadence. Confidence describes how much evidence supports the analytical state of that row.

A row can be on schedule and low-confidence. A row can be delayed and still mathematically valid as the latest available state.

Chain	Expected lag	Soft warning	Hard fail
Bitcoin	~1 day	> 2 days	> 4 days
Ethereum	~1 day	> 2 days	> 4 days
Arbitrum	~7 days	> 10 days	> 15 days
Base	~7 days	> 10 days	> 15 days

Freshness fields

Field	Meaning
`updated_through`	Latest Gold observation date available to the Meta calculation.
`lag_days_vs_asof_date`	Publication-time freshness relative to the row date.
`lag_days_vs_utc_today`	Runtime freshness relative to the current UTC date. This may differ between publication and reading time.

13. Release Integrity and Determinism

Determinism hash construction

Every named-state Meta row — STABLE, HEATING, CONGESTED, or CHEAP — carries `regime.determinism_hash`.

UNKNOWN/DEGRADED rows do not carry a determinism hash. Their public identity is anchored by public row fields listed below.

Hash payload

The determinism hash is constructed over the public named-state regime payload:

```
payload = {
    "asof_date": meta["regime"]["asof_date"],
    "chain": meta["chain"],
    "drivers": meta["regime"]["drivers"],
    "label": meta["regime"]["label"],
    "ruleset_id": meta["regime"]["ruleset_id"],
}

canonical = json.dumps(payload, sort_keys=True, separators=(",", ":"))
hash = hashlib.sha256(canonical.encode("utf-8")).hexdigest()[:12]
```

Given the same payload, the same 12-character string is produced. If the named regime payload changes, the hash changes.

The hash anchors the public named-state payload. It is not a claim that all upstream internals are disclosed.

Public row identity model

Row type	Public identity fields
All Meta rows	`chain`, `date`, `methodology_version`
Named-state rows	`chain`, `date`, `methodology_version`, `regime.determinism_hash`
UNKNOWN/DEGRADED rows	`chain`, `date`, `methodology_version`, `updated_through`, `confidence.confidence_score`, `status.label`

UNKNOWN/DEGRADED rows do not carry a determinism hash because the named-state regime payload is not meaningful in a degraded state. Drivers are empty and the label is determined by the confidence gate, not by axis structure.

14. Provenance, Versioning, and Archive Policy

Public artifacts should be interpreted under the methodology version that governed them when published.

Provenance concept	Public meaning	When it changes
`date`	UTC day the row describes.	Only if row identity changes.
`updated_through`	Most recent Gold observation used by the Meta calculation.	When the freshest supporting row changes.
`methodology_version`	Public methodology version governing interpretation.	When field meaning or analytical semantics change materially.
`regime.determinism_hash`	Named-row integrity anchor.	When the named regime payload changes.
Dataset / published revision	Internal publication-batch identity at the dataset layer. This is useful for release traceability but is not a required public join key and should not be treated as a separate revision integer in customer workflows.	When a new publication batch is issued.

Archived-as-published principle

Historical outputs are intended to remain interpretable under the methodology version under which they were published. If methodology changes in a way that changes field meaning or label semantics, that change is versioned rather than silently applied retroactively.

15. Verification and Evidence Paths

Urd Atlas separates public verification into three classes.

Class	Definition	Typical examples
A — Directly reproducible	Can be reproduced from published artifacts alone.	Determinism hash, confidence gate, scorecard display score, published label identity.
B — Independently checkable	Can be checked against public chain evidence but cannot be fully reconstructed from Urd Atlas artifacts alone.	Daily counts, block count, fee direction, block-time behavior.
C — Black box by design	Publicly interpretable but intentionally not fully reconstructable.	Private calibration constants, intermediate feature tables, exact upstream join logic.

Diligence boundary

Urd Atlas is designed to make published outputs auditable without pretending the entire production classifier is open-sourced. Reproducibility of the public trust layer and full disclosure of every proprietary internal are different concerns.

Intended diligence path

A technical buyer should be able to:

1. Load Gold for a chain/date.

2. Load Derived for the same chain/date.
3. Load Meta for the same chain/date.
4. Inspect confidence and gate threshold.
5. Verify named-state determinism hash where present.
6. Inspect drivers and current values.
7. Compare relevant Class B facts against public chain evidence.
8. Preserve the row with its `methodology_version`.

16. API Delivery Model

Urd Atlas is delivered primarily as JSON reference data.

The website presents documentation, examples, status, sample files, methodology, and product explanations. The subscriber-facing data product is accessed through authenticated API delivery.

Public sample access

The public sample pack exists for pre-purchase diligence, documentation, and integration testing. It exposes representative artifacts and verification examples. It is not a substitute for a live subscription, full archive, or current production feed.

Subscriber API

Subscriber API access is intended for machine-to-machine retrieval of published JSON artifacts.

Typical access patterns include:

- latest Meta for a chain,
- latest Gold and Derived files,
- historical files where entitlement permits,
- chain/date joins into user systems,
- periodic ingestion into internal analytics pipelines.

API key model

API keys are intended for subscriber file access. Full API secrets are shown only at creation time and should be stored securely by the customer. Urd Atlas should not be expected to recover a lost full API secret.

Fair use and quotas

API access is subject to technical rate limits, daily quotas, entitlement checks, abuse prevention, and fair-use terms. These controls protect platform reliability and ensure that one user cannot degrade service quality or create disproportionate operating cost.

Rate and quota controls are operational controls. They do not alter the meaning of published data.

Public sample route vs subscriber file delivery

Public samples and subscriber file delivery are intentionally separated. Public sample access is for limited, whitelisted example artifacts. Subscriber file delivery is authenticated and entitlement-checked.

Corrections and restatements

If a correction or restatement is issued for published artifacts, it should be disclosed through the public changelog, release notes, dataset metadata, or other customer-facing notice appropriate to the severity of the correction. Corrections should identify the affected chains, dates, artifact layers, and methodology impact. A correction does not automatically imply a methodology version bump; a bump is required when field meaning or analytical semantics change materially.

17. Operational Trust Controls

Operational trust is supported by controls around identity, payment, API access, publication integrity, and public verification.

Control	Public meaning
Identity/session layer	User sign-in and account sessions are handled by a third-party identity provider. As of this document version, Urd Atlas uses Clerk-class identity/session infrastructure. Provider changes do not require a methodology version bump if artifact semantics are unchanged.
Payment layer	Payment processing is handled by a third-party payment provider. As of this document version, Urd Atlas uses Stripe-class payment infrastructure. Urd Atlas does not need to store card numbers.
API keys	Subscriber API access uses non-recoverable API secrets.
Rate limits and quotas	API usage is limited to protect service reliability and platform cost.
Sample/public separation	Public examples are separated from authenticated subscriber delivery.
Versioned methodology	Artifact meaning is governed by `methodology_version`.
Public verification	Named-state hashes and sample-pack scripts support independent checking of public payload integrity.
Non-advisory boundary	Documentation and product copy explicitly exclude recommendations, forecasts, and price signals.

These controls do not make the system risk-free. They define the product's public trust posture and the operational boundaries a customer can evaluate.

18. Disclosure Boundaries

The public methodology is designed to make the product auditable in meaning without enabling source-data reconstruction or pipeline cloning.

Public methodology discloses	Public methodology intentionally does not disclose
Field meaning, artifact ownership, and data contract.	Exact upstream schemas or private source join logic.
Interpretation of confidence, scorecard, regime, and freshness.	Intermediate feature tables and ingestion repair rules.
Publicly relevant thresholds and gate values.	Enough detail to reconstruct raw source rows from published aggregates.
Worked verification examples.	Enough implementation detail to clone the private pipeline.
Classification rules, label order, and band thresholds.	Full proprietary calibration constants required for end-to-end private reproduction.
Freshness policy and chain-specific cadence.	Internal operational topology, secrets, credentials, or private infrastructure details.

This boundary is intentional. A reference data product can be auditable without being fully open-source.

19. Analytical Interpretation Boundaries

Boundary	Operational consequence
Chain-relative only	Labels and scores are relative to each chain's own historical distribution. CONGESTED on Arbitrum does not imply the same absolute state as CONGESTED on Ethereum.
Labels can change day to day	CONGESTED and CHEAP can fire from single-day threshold crossings. Consumers requiring multi-day stability must apply their own filters.
Scorecard and driver z-scores differ	They use different input series, windows, and normalization constants.
Bitcoin Capacity is an instability proxy	It measures deviation from recent block-time norm in either direction, not directional slow-block congestion.
Friction can rise without high nominal fees	`fee_burden_proxy` is a ratio. Friction may increase because transferred value is low, not because absolute fees are high.
No price, forecast, or recommendation	No field constitutes a price observation, return forecast, trading signal, or investment recommendation.
Briefs are explanatory	Briefs summarize published evidence; they do not override Meta or create an advisory layer.

20. Field Reference Summary

Key Gold fields

Field	Type	Meaning	Verification
`tx_count_daily`	integer	Confirmed daily transaction count.	B
`block_count_daily`	integer	Blocks produced on the UTC day.	B
`median_tx_fee_native`	float	Typical same-day fee in native denomination.	B
`median_tx_value_native`	float	Typical same-day transaction value, used in friction proxy.	B
`avg_block_time_sec`	float	Typical daily inter-block interval.	B
`gas_utilization_pct`	float/null	Gas utilization as fraction of block gas limit. Null for Bitcoin.	B
`failed_tx_rate`	float/null	Fraction of transactions that failed. Null where not semantically applicable.	B
`unique_active_addresses`	int/null	Participation breadth. Null where not reliably available.	B

Key Meta fields

Field	Type	Meaning	Verification
`status.label`	string	Top-line regime state.	A
`confidence.confidence_score`	float	Composite confidence [0,1].	A
`publish_confidence.threshold`	float	Canonical gate threshold.	A
`regime.determinism_hash`	string/absent	12-character SHA-256 prefix over public regime payload. Present on named-state rows only.	A
`regime.ruleset_id`	string	Chain-specific classification ruleset identifier.	A
`regime.drivers[]`	array	Ranked driver set with metric, axis, z-score, percentile, momentum, and current value.	A/C
`scorecard.dimensions[].score`	float	Confidence-degraded display score [0,100].	A
`methodology_version`	string	Methodology version governing interpretation.	A
`updated_through`	date	Latest Gold observation available to this Meta calculation.	A

21. Changelog and Versioning Policy

Date	Class	Affected	Methodology bump?	Historical rows changed?	Subscriber action?	Summary
2026-04-21	docs-only	Methodology hub	No	No	No	Introduced structured methodology hub and cross-linked trust-layer pages.
2026-04-21	interpretation-only	Reference / fields / boundaries	No	No	No	Clarified label volatility, scorecard vs driver normalization, fee-burden semantics, and BTC capacity as instability proxy.
2026-04-21	contract clarification	Schema / provenance docs	No	No	No	Removed public reliance on a required separate revision integer; aligned provenance with fields present in archive.
2026-04-21	operational docs	Service / API docs / pricing	No	No	No	Added service expectations, public sample pack, and common workflows for diligence.
2026-05-03	methodology consolidation	Whitepaper / public methodology	Yes v1.2	No	Yes — cite v1.2 going forward	Consolidated methodology, confidence, scorecard, regime rules, verification, provenance, and sample payload documentation into canonical whitepaper. Clarified UNKNOWN/DEGRADED hash behavior.
2026-05-25	docs/product clarification	Whitepaper / API delivery / sample access / operational trust controls	No	No	No	Clarified Regime Briefs as an explanatory publication layer, subscriber API delivery model, public sample access, fair-use controls, correction/restatement notice expectations, operating-entity disclosure boundary, and operational trust boundaries. No change to classification methodology, thresholds, confidence gate, or artifact semantics.

22. Conclusion

Urd Atlas is a deterministic regime reference data layer for on-chain network-state analysis. Its value is not that it replaces researcher judgment, but that it gives researchers a documented, versioned, auditable state variable that can be joined into their own systems.

The system is intentionally narrow. It does not publish price data, does not forecast, and does not recommend actions. It publishes descriptive classifications of chain-relative conditions and exposes enough evidence, confidence, freshness, provenance, and integrity metadata to support technical due diligence.

For a research-grade user, the critical product property is repeatability. A customer should be able to ingest a row, inspect the evidence surface, check the confidence gate, verify the public hash where present, and preserve the row under the methodology version that governed its interpretation.

Appendix A: Representative Sample Payloads

The following payloads are representative examples for documentation. Full sample files should be verified against the current public sample pack.

A.1 Gold layer — Ethereum


```
{
  "chain": "ethereum",
  "date": "2026-03-31",
  "updated_through": "2026-03-31",
```

```
  "tx_count_daily": 1423041,
  "median_tx_fee_native": 0.00342,
  "gas_utilization_pct": 0.962,
  "failed_tx_rate": 0.071,
  "avg_block_time_sec": 12.1,
  "block_count_daily": 7143
}
```


A.2 Derived layer — Ethereum


```
{
  "chain": "ethereum",
  "date": "2026-03-31",
  "updated_through": "2026-03-31",
  "tx_count_daily__ma7": 1382201.0,
  "tx_count_daily__ma30": 1284004.0,
  "median_tx_fee_native__ma7": 0.00311,
  "median_tx_fee_native__ma30": 0.00227,
  "gas_utilization_pct__ma7": 0.948,
  "gas_utilization_pct__ma30": 0.901
}
```


A.3 Meta layer — named state


```
{
  "chain": "ethereum",
  "date": "2026-03-31",
  "updated_through": "2026-03-31",
  "methodology_version": "v1.2",
  "status": {
    "label": "CONGESTED",
    "one_liner": "Demand, fees, and utilization remain elevated versus recent history."
  },
  "publish_confidence": {
    "threshold": 0.40
  },
  "confidence": {
    "confidence_score": 0.847,
    "data_quality_score": 0.91,
    "label_confidence_score": 0.79,
    "lag_days_vs_utc_today": 0
  },
  "regime": {
    "label": "CONGESTED",
    "asof_date": "2026-03-31",
    "ruleset_id": "eth_l1_v1",
    "window_days": 180,
    "determinism_hash": "81b295000696",
    "drivers": [
      {
        "axis": "demand",
        "metric": "tx_count_daily",
        "z_robust": 2.18,
        "pct_90d": 0.97,
```

```
        "momentum_7d_vs_30d": 0.44,
        "current": 1423041
      },
      {
        "axis": "friction",
        "metric": "median_tx_fee_native",
        "z_robust": 2.41,
        "pct_90d": 0.99,
        "momentum_7d_vs_30d": 0.58,
        "current": 0.00342
      },
      {
        "axis": "capacity",
        "metric": "gas_utilization_pct",
        "z_robust": 1.84,
        "pct_90d": 0.95,
        "momentum_7d_vs_30d": 0.21,
        "current": 0.962
      }
    ]
  }
}
```


A.4 Meta layer — UNKNOWN/DEGRADED

UNKNOWN/DEGRADED rows do not carry a determinism hash. Row identity is anchored by chain, date, methodology version, updated-through date, confidence score, and status label.


```
{
  "chain": "ethereum",
  "date": "2025-04-21",
  "updated_through": "2025-04-21",
  "methodology_version": "v1.2",
  "status": {
    "label": "UNKNOWN/DEGRADED",
    "one_liner": "Evidence quality is below the named-state publication threshold."
  },
  "publish_confidence": {
    "threshold": 0.40
  },
  "confidence": {
    "confidence_score": 0.318,
    "data_quality_score": 0.41,
    "label_confidence_score": 0.25,
    "lag_days_vs_utc_today": 8
  },
  "regime": {
    "label": "UNKNOWN/DEGRADED",
    "asof_date": "2025-04-21",
    "ruleset_id": "eth_l1_v1",
    "window_days": 180,
    "drivers": []
  }
}
```

A.5 Meta layer — HEATING


```
{
  "chain": "arbitrum",
  "date": "2026-03-25",
  "updated_through": "2026-03-25",
  "methodology_version": "v1.2",
  "status": {
    "label": "HEATING",
    "one_liner": "Activity is firming relative to recent chain norms but not yet congested."
  },
  "publish_confidence": {
    "threshold": 0.40
  },
  "confidence": {
    "confidence_score": 0.732,
    "data_quality_score": 0.81,
    "label_confidence_score": 0.66,
    "lag_days_vs_utc_today": 0
  },
  "regime": {
    "label": "HEATING",
    "asof_date": "2026-03-25",
    "ruleset_id": "arb_v1",
    "window_days": 180,
    "determinism_hash": "da96a075f205",
    "drivers": [
      {
        "axis": "demand",
        "metric": "tx_count_daily",
        "z_robust": 1.36,
        "pct_90d": 0.83,
        "momentum_7d_vs_30d": 0.17,
        "current": 1182403
      }
    ]
  }
}
```


Appendix B: Verification Script

The following Python script verifies the public named-state determinism hash pattern for a sample Meta file. It is parameterized so the sample-pack location can change without making the whitepaper obsolete.

By default, the script verifies self-consistency: it recomputes the determinism hash from the Meta payload and checks that the result matches the hash published in the file. If an expected hash is provided explicitly, the script also checks against that value.

```
import argparse
import hashlib
import json
from pathlib import Path
```

```
def load_json(path: Path) -> dict:
    return json.loads(path.read_text(encoding="utf-8"))


def determinism_hash(meta: dict) -> str:
    payload = {
        "asof_date": meta["regime"]["asof_date"],
        "chain": meta["chain"],
        "drivers": meta["regime"]["drivers"],
        "label": meta["regime"]["label"],
        "ruleset_id": meta["regime"]["ruleset_id"],
    }

    canonical = json.dumps(payload, sort_keys=True, separators=(",", ":"))
    return hashlib.sha256(canonical.encode("utf-8")).hexdigest()[:12]


def main() -> None:
    parser = argparse.ArgumentParser(
        description="Verify Urd Atlas sample-pack artifact consistency."
    )
    parser.add_argument(
        "--base",
        default="sample-pack/ethereum/2026-03-31",
        help="Directory containing gold.json, derived.json, and meta.json.",
    )
    parser.add_argument(
        "--expected-hash",
        default=None,
        help="Optional expected determinism hash for the sample Meta row.",
    )

    args = parser.parse_args()
    base = Path(args.base)

    gold = load_json(base / "gold.json")
    derived = load_json(base / "derived.json")
    meta = load_json(base / "meta.json")

    assert gold["chain"] == meta["chain"]
    assert derived["chain"] == meta["chain"]
    assert gold["date"] == meta["date"]
    assert derived["date"] == meta["date"]

    for driver in meta["regime"].get("drivers", []):
        metric = driver.get("metric")
        if metric in gold and "current" in driver:
            assert driver["current"] == gold[metric]

    required_derived_fields = [
        "tx_count_daily__ma7",
        "tx_count_daily__ma30",
        "median_tx_fee_native__ma7",
        "median_tx_fee_native__ma30",
```

```
    ]

    missing = [field for field in required_derived_fields if field not in derived]
    assert not missing, {"missing_derived_fields": missing}

    assert meta["publish_confidence"]["threshold"] == 0.40

    label = meta["regime"]["label"]

    if label == "UNKNOWN/DEGRADED":
        assert "determinism_hash" not in meta["regime"]
        print("UNKNOWN/DEGRADED row verified without determinism hash.")
    else:
        computed = determinism_hash(meta)
        published = meta["regime"]["determinism_hash"]

        assert computed == published, {
            "computed": computed,
            "published": published,
        }

        if args.expected_hash is not None:
            assert computed == args.expected_hash, {
                "computed": computed,
                "expected": args.expected_hash,
            }

        print("Named-state determinism hash verified.")
        print({"determinism_hash": computed})

    print(
        {
            "chain": meta["chain"],
            "date": meta["date"],
            "label": meta["status"]["label"],
            "gate_threshold": meta["publish_confidence"]["threshold"],
        }
    )


if __name__ == "__main__":
    main()
```

Appendix C: Glossary of Public Terms

Term	Definition
Gold	Daily observation layer. Direct chain metrics and robust daily summaries without regime logic, confidence gating, or categorical interpretation.
Derived	Deterministic rolling transforms of Gold metrics, primarily MA7 and MA30.
Meta	Classification layer containing regime label, confidence, scorecard, drivers, freshness, methodology version, and provenance fields.

Term	Definition
Regime Briefs	Explanatory summaries derived from the same evidence surface as the canonical artifacts. Briefs are descriptive, not advisory.
Named-state row	A Meta row whose label is STABLE, HEATING, CONGESTED, or CHEAP. Named-state rows carry a determinism hash.
UNKNOWN/DEGRADED	A Meta row where confidence is below the publication gate. No determinism hash is issued.
Confidence gate	The publication threshold that blocks named labels when composite confidence is too low.
Determinism hash	A 12-character SHA-256 prefix over the canonical public regime payload for named-state rows.
Ruleset ID	Public identifier tying a row to a chain-specific classification ruleset.
Chain-relative	Scores and labels are interpreted relative to the chain's own historical distribution.
`fee_burden_proxy`	`median_tx_fee_native / median_tx_value_native`. A dimensionless ratio measuring fee cost relative to transferred value.
`blocktime_instability`	Bitcoin-specific capacity proxy measuring rolling deviation from recent block-time norm.
Verification Class A	Directly reproducible from published artifacts alone.
Verification Class B	Independently checkable against public chain evidence but not fully reconstructable from Urd Atlas artifacts alone.
Verification Class C	Publicly interpretable but intentionally not fully reconstructable to prevent source-data reconstruction or pipeline cloning.
Archived-as-published	Historical artifacts are retained as originally published unless an explicit correction is issued and documented.

End of document. For the latest version of this whitepaper and full sample-pack access, visit urdatlas.com. Technical questions: support@urdatlas.com.