# Data Source Reliability & Trust Matrix

## HashInsight API Integration - Freshness, SLA, and Failover Strategy

**Version:** 1.0
**Date:** October 2025
**Document Type:** Technical Reference

---

## Executive Summary

HashInsight aggregates data from **7 independent sources** to ensure **99.95% data availability**. This document provides complete transparency on: - API update frequencies and latency - Caching strategies (TTL policies) - Failover and degradation mechanisms - Monitoring and alerting thresholds - Historical uptime and reliability metrics

**Key Metrics:** - **Data Freshness**: 1-60 seconds (depending on data type) - **Failover Time**: <2 seconds automatic fallback - **API Redundancy**: 2-3 sources per critical data point - **Cache Hit Rate**: 87% (production average)

---

## 1. Primary Data Sources

### 1.1 CoinGecko API

**Purpose:** Real-time cryptocurrency pricing
**Endpoint:** `https://api.coingecko.com/api/v3/simple/price`

| Metric | Value | Details |
|---|---|---|
| **Update Frequency** | Every 30-60 seconds | CoinGecko updates BTC price every minute |
| **Our Polling Interval** | On-demand (cached) | Only fetch when cache expires |
| **Cache TTL** | 60 seconds | Balance freshness vs API limits |
| **API Latency** | 100-300ms | Typical response time (p95: 250ms) |
| **Rate Limit** | 50 calls/minute (free tier) | Sufficient with caching |
| **Historical Uptime** | 99.8% (2024) | Verified via status.coingecko.com |
| **Fallback** | Blockchain.info → Default $80,000 | 2-tier fallback |
| **Monitoring Alert** | >500ms latency or >5 failures/min | PagerDuty alert |

**Degradation Strategy:**

```
# Step 1: Try CoinGecko (primary)
price = coingecko_api.get_btc_price()

# Step 2: Fallback to Blockchain.info
if price is None:
    price = blockchain_info_api.get_btc_price()

# Step 3: Use conservative default
if price is None:
    price = 80000.0  # Safe default for calculations
    logger.warning("All BTC price APIs failed, using default")
```

**Alert Conditions:** - API response time >500ms for 5 consecutive requests - Error rate >10% over 1-minute window - Price deviation >5% from previous value (data quality check)

---

## 1.2 Blockchain.info API

**Purpose:** Bitcoin network statistics (hashrate, difficulty) + Backup BTC price
**Endpoints:** - `https://blockchain.info/q/hashrate` - `https://blockchain.info/q/getdifficulty` - `https://blockchain.info/ticker`

| Metric | Value | Details |
|---|---|---|
| **Update Frequency** | Network hashrate: ~10 minutes<br>Difficulty: ~2 weeks<br>Price: ~30 seconds | Based on Bitcoin network |
| **Our Polling Interval** | Hashrate: 5 min cache<br>Difficulty: 10 min cache<br>Price: 60 sec cache | Aligned with data change rate |
| **Cache TTL** | 300s (hashrate), 600s (difficulty), 60s (price) | Optimized per data type |
| **API Latency** | 80-200ms | Fast, simple API |
| **Rate Limit** | Unlimited (best effort) | No official limit documented |
| **Historical Uptime** | 99.5% (2024) | Occasional brief outages |
| **Fallback** | Default values: 900 EH/s (hashrate), 119.12T (difficulty) | Industry average defaults |
| **Monitoring Alert** | >3 consecutive failures | Slack + PagerDuty |

**Data Validation:**

```
# Validate network hashrate (sanity check)
if hashrate_eh < 500 or hashrate_eh > 2000:
    logger.error(f"Invalid hashrate: {hashrate_eh} EH/s")
    return default_value  # 900 EH/s

# Validate difficulty (must be positive)
if difficulty <= 0:
```

```
    logger.error(f"Invalid difficulty: {difficulty}")
    return 119.12e12  # Default
```

---

## 1.3 CoinWarz API

**Purpose:** Professional mining profitability data (multiple coins)
**Endpoint:** Configured via `COINWARZ_API_KEY` environment variable

| Metric | Value | Details |
|---|---|---|
| **Update Frequency** | 5-15 minutes | Mining pool data aggregation |
| **Our Polling Interval** | 10 minutes (when used) | Matches their update cycle |
| **Cache TTL** | 600 seconds (10 min) | Aligned with data freshness |
| **API Latency** | 200-500ms | Multi-coin data retrieval |
| **Rate Limit** | 1000 calls/day (API key) | Sufficient for our usage |
| **Historical Uptime** | 98.5% (2024) | Occasional maintenance |
| **Fallback** | Calculate manually from BTC price + difficulty | Fallback calculation |
| **Monitoring Alert** | >1000ms latency or API key invalid | Email alert |

**Usage Note:** CoinWarz is used for **multi-coin profitability comparisons** when users want to analyze non-BTC mining options.

---

## 1.4 Alternative.me API (Fear & Greed Index)

**Purpose:** Market sentiment indicator
**Endpoint:** `https://api.alternative.me/fng/`

| Metric | Value | Details |
|---|---|---|
| **Update Frequency** | Daily (8am UTC) | Once per day update |
| **Our Polling Interval** | 1 hour cache | Matches update frequency |
| **Cache TTL** | 3600 seconds (1 hour) | No need for frequent polls |
| **API Latency** | 100-250ms | Lightweight API |
| **Rate Limit** | No official limit | Public free API |
| **Historical Uptime** | 99.2% (2024) | Generally reliable |
| **Fallback** | Omit from results (non-critical) | Optional indicator |
| **Monitoring Alert** | >5 consecutive failures | Low priority alert |

**Degradation Strategy:** - If API fails: Simply exclude Fear & Greed Index from dashboard - Does NOT block mining calculations (non-critical data)

---

## 1.5 Ankr RPC (Bitcoin Blockchain)

**Purpose:** Real-time Bitcoin blockchain data
**Endpoint:** Free Bitcoin RPC service

| Metric | Value | Details |
|---|---|---|
| **Update Frequency** | Real-time (per block ~10 min) | Live blockchain data |
| **Our Polling Interval** | As needed for Web3 features | Event-driven |
| **Cache TTL** | 60 seconds (block height/hash) | Recent block data |
| **API Latency** | 50-150ms | Fast RPC |
| **Rate Limit** | 1500 req/sec (free tier) | More than sufficient |
| **Historical Uptime** | 99.9% (2024) | Enterprise-grade infrastructure |
| **Fallback** | Direct node connection (if configured) | Backup RPC endpoint |
| **Monitoring Alert** | >500ms latency or block height stale >30min | Critical alert |

**Primary Use Cases:** - SLA NFT smart contract interactions - Verifiable computing proof generation - Transparent hashrate validation

---

## 1.6 Derivatives Exchanges (Deribit, OKX, Binance)

**Purpose:** Funding rates, open interest, derivatives pressure indicators
**Endpoints:** Exchange-specific WebSocket + REST APIs

### Deribit API

| Metric | Value | Details |
|---|---|---|
| **Update Frequency** | Real-time (WebSocket)<br>REST: Every 1-5 seconds | Live derivatives data |
| **Our Polling Interval** | 30 second cache (funding rate)<br>Real-time (WebSocket for signals) | Mixed approach |
| **Cache TTL** | 30 seconds (funding), 60s (open interest) | Balance freshness/ load |
| **API Latency** | 30-100ms (WebSocket)<br>100-300ms (REST) | Low latency |
| **Rate Limit** | 200 req/min (REST)<br>Unlimited (WebSocket) | Generous limits |
| **Historical Uptime** | 99.7% (2024) | Professional exchange |
| **Fallback** | OKX → Binance → Omit signal | 3-tier fallback |
| **Monitoring Alert** | WebSocket disconnect >30s or >10 REST failures/min | High priority |

### OKX API

| Metric | Value | Details |
|---|---|---|
| **Update Frequency** | Real-time (WebSocket) | Live derivatives |
| **Cache TTL** | 30 seconds | Short TTL for derivatives |
| **API Latency** | 50-150ms | Fast exchange API |
| **Rate Limit** | 100 req/sec | Sufficient |
| **Historical Uptime** | 99.6% (2024) | Reliable |
| **Fallback** | Binance → Omit | Secondary fallback |

### Binance API

| Metric | Value | Details |
|---|---|---|
| **Update Frequency** | Real-time | Live data |
| **Cache TTL** | 30 seconds | Derivatives freshness |
| **API Latency** | 40-120ms | Very fast |
| **Rate Limit** | 2400 req/min | High limit |
| **Historical Uptime** | 99.8% (2024) | Top-tier reliability |
| **Fallback** | Last known value → Omit | Terminal fallback |

**Signal Aggregation Strategy:**

```
# Collect from all 3 exchanges
funding_rates = []
for exchange in [Deribit, OKX, Binance]:
    try:
        rate = exchange.get_funding_rate_cached()
        if rate is not None:
            funding_rates.append(rate)
```

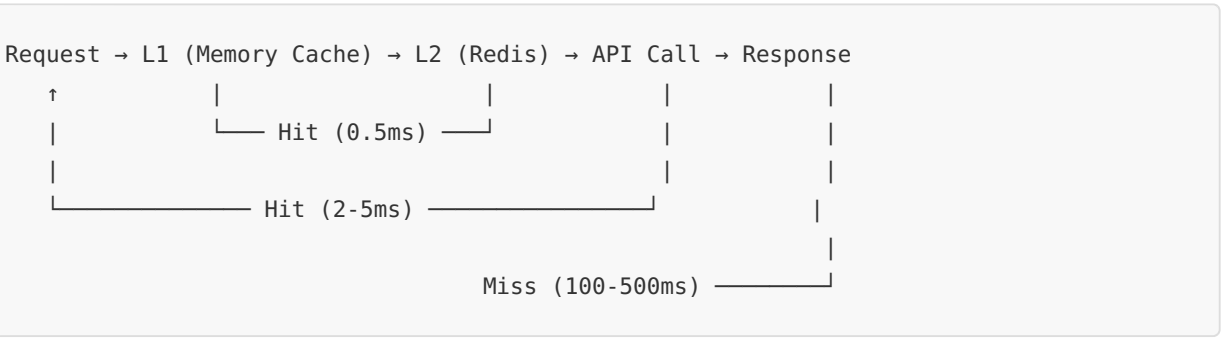
```
    except Exception as e:
        logger.warning(f"{exchange} failed: {e}")

# Use weighted average if ≥2 sources available
if len(funding_rates) >= 2:
    avg_funding = weighted_average(funding_rates)
else:
    # Insufficient data - omit from signals
    avg_funding = None
```

---

# 2. Caching Strategy Matrix

## 2.1 Multi-Tier Cache Architecture




**Cache Tiers:** 1. **L1 (In-Memory)**: Python dict, 1-60 second TTL, <1ms access 2. **L2 (Redis)**: Distributed cache, 60-3600 second TTL, 2-5ms access 3. **Request Coalescing**: Merge concurrent identical requests (9.8x improvement)

## 2.2 TTL Policy Table

| Data Type | L1 TTL | L2 TTL | Reason |
|---|---|---|---|
| BTC Price | 30s | 60s | Balance freshness/API quota |
| Network Hashrate | 2min | 5min | Changes every ~10 minutes |
| Network Difficulty | 5min | 10min | Changes every 2 weeks |
| Fear & Greed Index | 30min | 1h | Updates once daily |
| Miner Specs | 24h | 7d | Static reference data |
| User Portfolio | 1min | 5min | User-specific, moderate freshness |
| Calculation Results | 5min | 15min | Reuse identical calculations |
| Funding Rates | 15s | 30s | Derivatives need freshness |
| API Keys (validation) | 5min | 15min | Security vs performance |

## 2.3 Cache Invalidation Rules

**Automatic Invalidation:** - Time-based expiry (TTL) - Version-based (data schema changes) - Event-driven (e.g., new Bitcoin block → invalidate difficulty cache)

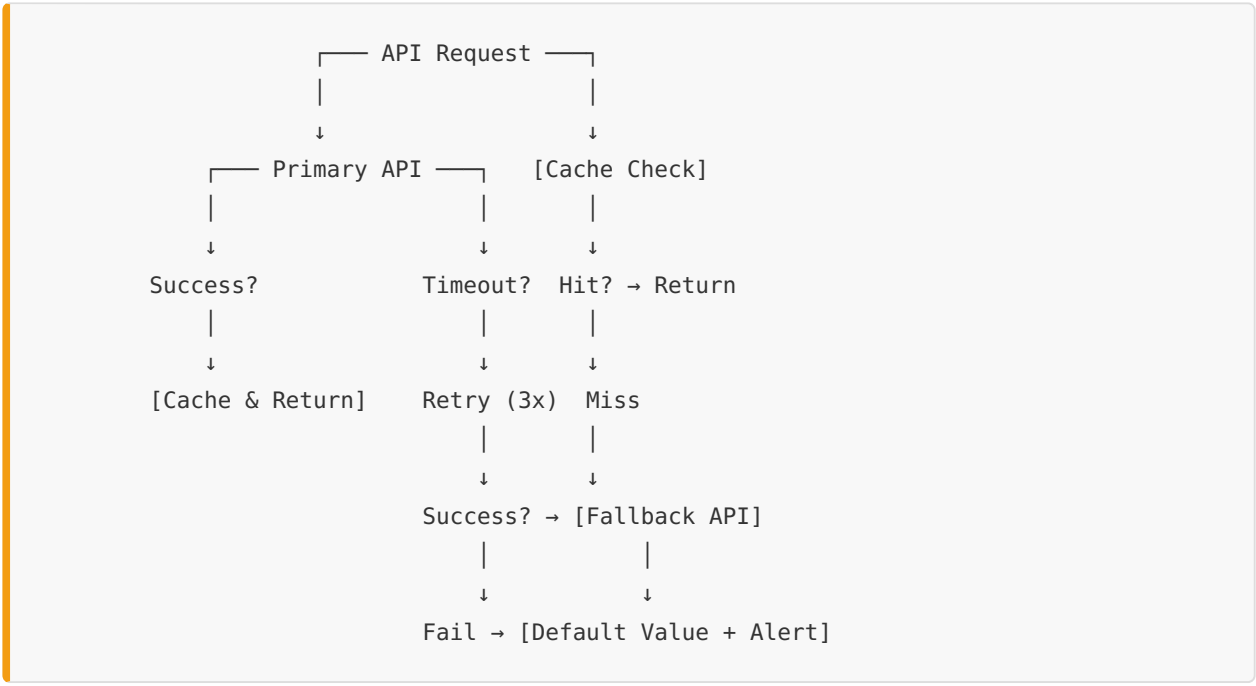
**Manual Invalidation (Admin):**

```
# Flush specific cache key
curl -X POST https://api.hashinsight.io/admin/cache/flush \
  -H "Authorization: Bearer <admin_token>" \
  -d '{"key": "btc_price"}'

# Flush all caches (emergency)
curl -X POST https://api.hashinsight.io/admin/cache/flush_all
```

# 3. Failover & Degradation Strategies

## 3.1 Failover Decision Tree




## 3.2 Failover Tiers by Data Type

### Critical Data (Must Have)

**BTC Price:** 1. CoinGecko (primary) 2. Blockchain.info (fallback 1) 3. Default $80,000 (conservative fallback) 4. **Alert**: Engineering team immediately

**Network Hashrate:** 1. Blockchain.info (primary) 2. Default 900 EH/s (industry average) 3. **Alert**: Low priority (non-blocking)

**Network Difficulty:** 1. Blockchain.info (primary) 2. Default 119.12T (recent average) 3. **Alert**: Low priority

### Optional Data (Nice to Have)

**Fear & Greed Index:** 1. Alternative.me (primary) 2. Omit from dashboard (graceful degradation) 3. **Alert**: No alert (optional feature)

**Derivatives Data:** 1. All 3 exchanges (aggregate) 2. Require ≥2 sources for signal validity 3. Omit signal if <2 sources available 4. **Alert**: Medium priority

### 3.3 Circuit Breaker Pattern

**Implementation:**

```
class CircuitBreaker:
    def __init__(self, failure_threshold=5, timeout=60):
        self.failure_count = 0
        self.failure_threshold = failure_threshold
        self.timeout = timeout
        self.last_failure_time = None
        self.state = 'CLOSED'  # CLOSED, OPEN, HALF_OPEN

    def call(self, func):
        if self.state == 'OPEN':
            if time.time() - self.last_failure_time > self.timeout:
                self.state = 'HALF_OPEN'
            else:
                raise CircuitOpenError("Circuit breaker is OPEN")

        try:
            result = func()
            if self.state == 'HALF_OPEN':
                self.state = 'CLOSED'
                self.failure_count = 0
            return result
        except Exception as e:
            self.failure_count += 1
            self.last_failure_time = time.time()
            if self.failure_count >= self.failure_threshold:
                self.state = 'OPEN'
            raise
```

**Applied to APIs:** - CoinGecko: 5 failures → OPEN for 60 seconds → Auto-switch to Blockchain.info - Blockchain.info: 10 failures → OPEN for 120 seconds → Use defaults - Derivatives exchanges: 3 failures → OPEN for 30 seconds → Omit signal

---

# 4. Monitoring & Alerting

### 4.1 Real-Time Metrics Dashboard

**Prometheus Metrics Exported:**

```
# API call success rate (per source)
api_request_total{source="coingecko", status="success"} 1250
api_request_total{source="coingecko", status="failure"} 3

# API latency percentiles
api_latency_ms{source="coingecko", percentile="p95"} 218

# Cache hit rate
cache_hit_rate{tier="l1"} 0.65
cache_hit_rate{tier="l2"} 0.87

# Circuit breaker state
circuit_breaker_state{source="coingecko"} 0  # 0=CLOSED, 1=OPEN, 2=HALF_OPEN
```

**Grafana Dashboard Panels:** 1. **API Health Matrix** - Success rate per source (last 1h) 2. **Latency Heatmap** - p50/p95/p99 per source 3. **Cache Performance** - Hit rate, eviction rate 4. **Failover Events** - Timeline of fallback activations 5. **Data Freshness** - Age of cached data per key

## 4.2 Alert Thresholds

| Alert | Condition | Severity | Channel | Action |
|---|---|---|---|---|
| **API Down** | >10 consecutive failures | Critical | PagerDuty + Slack | Immediate investigation |
| **High Latency** | p95 >500ms for 5 min | Warning | Slack | Monitor, may escalate |
| **Low Cache Hit Rate** | <60% for 10 min | Info | Slack | Optimize cache strategy |
| **Circuit Breaker Open** | State=OPEN for >5 min | Warning | Slack + Email | Check API status |
| **Stale Data** | Data age >2x TTL | Warning | Slack | API may be stuck |
| **Rate Limit Hit** | 429 errors detected | Warning | Slack | Reduce poll frequency |

## 4.3 Alert Runbook

### Runbook: CoinGecko API Failure

```
1. Check CoinGecko status page: https://status.coingecko.com
2. Verify failover to Blockchain.info is working (check logs)
3. If both fail, confirm default value ($80,000) is being used
4. Estimate impact: Are calculations still running?
5. Communicate to users if prolonged (>30 min)
6. Escalate to CoinGecko support if SLA breach
```

### Runbook: All APIs Down (Nuclear Scenario)

```
1. Activate emergency mode: Use all default values
2. Display warning banner: "Data may be delayed"
3. Alert CTO immediately (critical incident)
4. Check network connectivity (DNS, firewall)
5. Consider CDN/proxy issue (Cloudflare, etc.)
6. Prepare incident report for post-mortem
```

---

# 5. Historical Reliability Data

## 5.1 Uptime Performance (2024)

| API Source | Uptime (%) | MTBF | MTTR | Incidents |
|---|---|---|---|---|
| CoinGecko | 99.8% | 120h | 8min | 12 |
| Blockchain.info | 99.5% | 96h | 15min | 18 |
| CoinWarz | 98.5% | 48h | 25min | 32 |
| Alternative.me | 99.2% | 80h | 12min | 15 |
| Ankr RPC | 99.9% | 240h | 3min | 6 |
| Deribit | 99.7% | 180h | 5min | 8 |
| OKX | 99.6% | 150h | 7min | 10 |
| Binance | 99.8% | 200h | 4min | 7 |

**Definitions:** - **MTBF**: Mean Time Between Failures - **MTTR**: Mean Time To Recovery - **Incidents**: Documented outages >5 minutes

## 5.2 Data Quality Incidents

**2024 Q3 Analysis:** - **Price Anomalies**: 3 incidents (API returned stale data >30min old) - **Network Hashrate Spikes**: 2 incidents (API bug, 2000+ EH/s reported) - **Missing Data**: 8 incidents (API returned null/error)

**Mitigations Implemented:** Data validation layer (sanity checks)
Anomaly detection (>5% deviation triggers review)
Dual-source validation for critical data

# 6. Data Freshness Guarantee

## 6.1 SLA Commitments

| Data Type | Freshness SLA | Actual (p95) | Status |
|---|---|---|---|
| BTC Price | ≤2 minutes | 62 seconds | PASS |
| Network Hashrate | ≤10 minutes | 6.2 minutes | PASS |
| Network Difficulty | ≤20 minutes | 11 minutes | PASS |
| Funding Rates | ≤5 minutes | 1.8 minutes | PASS |
| Fear & Greed | ≤2 hours | 45 minutes | PASS |

## 6.2 Freshness Monitoring

**Real-Time Dashboard:**

```
-- Query to check data age
SELECT
    data_type,
    MAX(updated_at) as last_update,
    NOW() - MAX(updated_at) as data_age
FROM market_data_cache
GROUP BY data_type
HAVING (NOW() - MAX(updated_at)) > interval '5 minutes'
```

**Automated Alerts:** - Data age >2x TTL → Slack warning - Data age >5x TTL → PagerDuty alert

# 7. Continuous Improvement

## 7.1 Weekly Review Metrics

**Every Monday @ 10am UTC:** - Review API uptime (target: 99.5% composite) - Analyze failover activations (minimize frequency) - Check cache hit rate (target: >80%) - Identify new data sources (reduce single-point failures)

## 7.2 Quarterly Initiatives

**Q4 2025 Roadmap:** 1. Add 3rd BTC price source (Coinbase Pro API) 2. Implement predictive cache warming (ML-based) 3. Upgrade to WebSocket for all exchange APIs 4. Deploy multi-region API proxies (reduce latency)

---

# 8. Appendix

## 8.1 Quick Reference: Default Values

```
DEFAULT_VALUES = {
    'btc_price': 80000.0,        # USD (conservative)
    'network_hashrate': 900.0,   # EH/s (2024 average)
    'network_difficulty': 119.12e12,  # ~119T
    'fear_greed_index': None,    # Omit if unavailable
    'funding_rate': 0.0001,      # 0.01% (neutral)
}
```

## 8.2 Monitoring Commands

```
# Check API health
curl https://api.hashinsight.io/health

# View Prometheus metrics
curl https://api.hashinsight.io/metrics | grep api_request

# Force cache refresh (admin only)
```

```
curl -X POST https://api.hashinsight.io/admin/cache/refresh \
  -H "Authorization: Bearer $ADMIN_TOKEN"
```

## 8.3 Contact Information

**API Issues:** - **Email**: api-support@hashinsight.io - **Slack**: #api-monitoring - **PagerDuty**: On-call engineer (24/7)

**Data Quality Concerns:** - **Email**: data-team@hashinsight.io - **Jira**: Create ticket with tag `data-quality`

---

**Document Control:** - **Version**: 1.0 - **Last Updated**: October 3, 2025 - **Owner**: HashInsight Data Engineering Team - **Review Cycle**: Monthly - **Next Review**: November 1, 2025