

ChainFeed Developer Guide

Version: 0.9

Date: 2025-10-19

Maintainer: Ernie Varitimos (FatTail Systems)

Repository: github.com/dudefromearth/ChainFeed

1. Introduction

This guide provides developers with everything needed to understand, modify, and extend the **ChainFeed** system — the real-time data backbone for the FatTail AI and Options Intelligence ecosystem.

ChainFeed ingests, normalizes, and publishes options chain data into Redis for downstream consumption (AI, SSE Gateway, Web Frontend).

The system follows a modular, antifragile design that prioritizes **observability, testability, and composability**.

2. Repository Layout

ChainFeed/

```
|— core/
|   |— chain_feed.py
|   |— chain_ingestor.py
|   |— chain_normalizer.py
|   |— historical_feed_manager.py
|   |— providers/
|       |— historical_provider.py
|       |— live_provider.py
|       |— synthetic_snapshot_provider.py
|
|— config/
```

```
| |—— chainfeed_constants.py
| |—— variant_config.yaml
| |—— groups.yaml
|
|—— utils/
| |—— redis_client.py
| |—— redis_inspect.py
| |—— heartbeat_watcher.py
| |—— expiration_utils.py
|
|—— test/
| |—— test_historical_provider.py
| |—— test_historical_ingest.py
| |—— test_normalize_snapshot.py
| |—— ...
|
|—— data/
| |—— formatted.json
| |—— (sample ES, SPY, etc.)
|
|—— notebooks/
| |—— notebook_heartbeat_analysis.py
|
|—— Makefile
```

3. Development Environment Setup

3.1 Prerequisites

- **Python 3.9+**
- **Redis (local instance)** running on `localhost:6379`
- **PyCharm** or **VSCode** recommended for development
- (Optional) Polygon API key for live or historical fetches

3.2 Initial Setup

```
git clone https://github.com/dudefromearth/ChainFeed.git
cd ChainFeed
python3 -m venv .venv
source .venv/bin/activate
pip install -r requirements.txt
```

Verify Redis connection:

```
redis-cli ping
# PONG
```

4. Core Concepts

4.1 Chain Feed Flow

1. Load source snapshot (local JSON or API call)
2. Normalize via ChainIngestor
3. Publish canonical snapshot to Redis
4. Emit group heartbeat with TTL
5. Monitor via watcher or analytics tool

4.2 Data Model

- **Snapshots** contain full options chain states
 - **Heartbeats** represent system-level liveness per group
 - **Groups** (e.g. `spx_complex`, `ndx_complex`) bundle correlated instruments (SPX, ES, SPY)
-

5. Key Components

5.1 Providers

Located in `core/providers/`.

HistoricalSnapshotProvider

- Loads data from file or API
- Returns normalized dictionary of contracts

Usage:

```
from core.providers.historical_provider import HistoricalSnapshotProvider
provider = HistoricalSnapshotProvider("SPX")
snapshot = provider.load_snapshot("data/formatted.json")
```

5.2 Chain Ingestor

Located in `core/chain_ingestor.py`.

Normalizes snapshot data and enforces structure.

```
python -m core.historical_feed_manager
```

Config-driven execution using `config/groups.yaml`.

5.4 Redis Integration

Simple, robust publishing to Redis:

```
r = redis.Redis(host="localhost", port=6379, db=0)
r.set("chain:spx_complex:SPX:snapshot", json.dumps(snapshot))
```

Verification:

```
python -m utils.redis_inspect
```

5.5 Heartbeat Watcher

Live terminal monitor for group health.

```
python -m utils.heartbeat_watcher
```

Displays:

- Time since last update
- TTL countdown
- Status transitions (Active → Overdue → Silent)

6. Testing & Validation

6.1 Running Tests

All unit tests use pytest:

```
pytest -v
```

6.2 Key Tests

| Test File | Purpose |
|-----------------------------|----------------------------------|
| test_historical_provider.py | Snapshot loading from local data |
| test_normalize_snapshot.py | Chain normalization verification |
| test_historical_ingest.py | Integration test for ingest flow |
| utils/redis_inspect.py | Redis snapshot validation |

7. Development Workflow

7.1 Daily Flow

1. Start Redis locally
2. Run `historical_feed_manager` to publish
3. Verify with `redis_inspect` and `heartbeat_watcher`
4. Write or adjust tests
5. Commit and push via PyCharm

7.2 Git Flow in PyCharm

1. **Git → Commit...** — select modified files
2. Add a message like:

Added SPX normalization and heartbeat watcher

3. **Commit and Push** to main

(Shortcut: `⌘K` then `⌘⇧K`)

8. Adding a New Group

To add a new market complex:

1. Edit config/groups.yaml

```
- name: Treasury Complex
  key: tsy_complex
  members:
    - symbol: ZB
      source_path: data/formatted_ZB.json
    - symbol: ZN
      source_path: data/formatted_ZN.json
```

2. Run:

```
python -m core.historical_feed_manager
```

3. Verify via:

```
python -m utils.redis_inspect
```


9. Observability & Analytics

9.1 Heartbeat Watcher

Real-time system health display:

💖 ChainFeed Heartbeat Monitor

💖 SPX_COMPLEX (heartbeat:spx_complex)

- Symbols: SPX
- TTL: 58s
- Status:  ACTIVE

9.2 Heartbeat Analysis Notebook

For historical reliability analysis:

```
python -m notebooks.notebook_heartbeat_analysis
```

Plots uptime ratios, transition maps, and (future) reliability metrics.

10. Extending ChainFeed

| Extension | Description |
|--------------------|--|
| Live Feed | Replace HistoricalProvider with real-time websocket ingestion. |
| Synthetic Feed | Create simulated option chains for testing models. |
| Cross-Asset Engine | Compute correlation and arbitrage between complexes. |
| Persistence Layer | Store all snapshots in PostgreSQL or Parquet. |
| SSE Gateway | Serve real-time chains to frontend via Server-Sent Events. |

11. Troubleshooting

| Symptom | Likely Cause | Fix |
|--|---------------------------------|--|
| FileNotFoundError: groups.yaml | Running from wrong directory | Run from project root (ChainFeed/) |
| Failed to publish snapshot: list has no attribute get | Invalid snapshot schema | Ensure file includes "contracts" list |

| | | |
|--------------------------|--------------------------|---|
| No heartbeat keys found | Feed manager not running | Run python -m core.historical_feed_manage r |
| Redis connection refused | Redis server not active | Run redis-server locally |

12. Contribution Guidelines

1. Branch from main
2. Follow PEP-8 and use type hints
3. Include minimal unit tests for all new features
4. Document new configs or constants
5. Submit pull request with concise commit history

13. Roadmap

| Milestone | Target | Description |
|-----------|---------|--|
| v1.0 | Q4 2025 | Live SPX & NDX feeds, Redis→SSE gateway |
| v1.1 | Q1 2026 | Synthetic + Statistical Arbitrage engine |
| v1.2 | Q2 2026 | Historical persistence + visualization |
| v1.3 | Q3 2026 | AI-driven market narrative generation |

14. License

All code © 2025 FatTail Systems.

Usage and redistribution are restricted under the FatTail Developer Agreement.