Upstox Intraday Momentum Algo – Conversation & Setup Notes

Saved: 2026-01-02T14:05:53.262398 (IST)

This document captures the outcome of our conversation and provides a consolidated, ready■to■run guide for the Python algo built for **Upstox** with:

- Segments: **NSE EQ, F&O;, Nifty & BankNIFTY options**
- Data mode: **LTPC** via **WebSocket V3** (binary Protobuf)
- Strategy: **Intraday momentum** (EMA■based on 1■minute bars)
- Risk: **Max drawdown 4%**, **1% per trade**
- Exits: **SL/TP**—either **GTT multi■leg** or **exchange SL/SL■M + LIMIT TP** with OCO via Portfolio Stream
- Auto **square■off @ 15:12 IST**
- Modes: **Paper**, **Backtest**, **Live**

Repository Structure (generated)

```
upx_algo/
■■ proto/                        # Protobufs (MarketDataFeedV3.proto + generated *_pb2.py)
■■ decoders/market_v3.py         # Protobuf decoder (LTPC)
■■ examples/query_chain.py       # BANKNIFTY option chain sample
■■ tools/
■  ■■ check_market_v3_decode.py # LTPC sanity test
■  ■■ mode_switch.py            # PAPER / LIVE (sandbox/real) toggle CLI
■  ■■ oauth_get_token.py        # OAuth helper (capture code, token exchange, update .env)
■  ■■ bod_refresh.py            # Daily BOD loader + ATM selection; updates .env
■  ■■ validate_post_refresh.py  # Post■refresh validator; emits JSON summary
■  ■■ schedule_bod.sh           # (Option A) cron wrapper for daily refresh
■  ■■ bod_refresh.ps1           # (Option C) Windows PowerShell wrapper
■  ■■ register_bod_task.ps1     # (Option C) Register Windows Scheduled Task (PowerShell)
■  ■■ register_bod_task.bat     # (Option C) Register Scheduled Task via schtasks (cmd)
■■ ops/systemd/                 # (Option B) systemd service & timer templates
■  ■■ upx-bod-refresh.service   # edit absolute paths; oneshot service
■  ■■ upx-bod-refresh.timer     # Mon–Fri 08:50 IST
■■ data/                        # BOD cache, validation JSON, CI sample
■  ■■ sample_instruments.json   # CI fallback sample instruments
■■ logs/                        # Rotating logs
■■ .github/workflows/upx-ci.yml # GitHub Actions: daily refresh & validation
■■ config.py                    # Loads .env
■■ auth.py                      # OAuth/token helper
■■ instrument_loader.py         # BOD loader + indexes
■■ enforce.py                   # lot/tick enforcement
■■ market_data.py               # WS V3 LTPC stream + bar aggregator
■■ portfolio_stream.py          # WS portfolio updates
■■ strategy_momentum.py         # EMA momentum signals
■■ risk.py                      # drawdown & sizing (1% per trade)
■■ broker_live.py               # OrderApiV3
■■ broker_paper.py              # paper simulator
■■ sl_tp_manager.py             # SL/SL■M + LIMIT TP, OCO via portfolio
■■ gtt_manager.py               # Bracket via GTT multi■leg
■■ engine.py                    # Orchestrator + square■off @ 15:12 IST
■■ main.py                      # Entrypoint
■■ requirements.txt             # Dependencies
■■ README.md                    # Quick start & references
■■ .env.sample                  # Sample environment config
```

Quick Start

```
# 1) Create venv
python -m venv .venv
source .venv/bin/activate

# 2) Install deps
pip install -r upx_algo/requirements.txt

# 3) Compile Protobuf classes for Market Data V3
#   Place MarketDataFeedV3.proto in upx_algo/proto (download from Upstox examples)
protoc --python_out=upx_algo/proto upx_algo/proto/MarketDataFeedV3.proto

# 4) Configure environment
cp upx_algo/.env.sample upx_algo/.env
# Edit UPX_API_KEY, UPX_API_SECRET, UPX_REDIRECT_URI

# 5) Obtain access token (OAuth)
python upx_algo/tools/oauth_get_token.py --open

# 6) Refresh BOD instruments & select ATM
python upx_algo/tools/bod_refresh.py --download --update-env --select atm --count 1

# 7) Validate
python upx_algo/tools/validate_post_refresh.py

# 8) Run (PAPER)
python upx_algo/tools/mode_switch.py --target paper
python upx_algo/main.py

# 9) LIVE (sandbox), then LIVE (real)
python upx_algo/tools/mode_switch.py --target live-sandbox
python upx_algo/main.py
python upx_algo/tools/mode_switch.py --target live-real
python upx_algo/main.py
```

OAuth Helper

Use `tools/oauth_get_token.py` to:

- Build the Upstox authorization URL (`response_type=code`)
- Open your browser for login & consent
- Capture the `code` via a tiny local HTTP server (e.g., `http://127.0.0.1:5000/callback`)
- Exchange the `code` for `access_token` via SDK, then update `.env`

> Ensure your developer app’s **Redirect URI** exactly matches the `.env` value.

Daily BOD Refresh & Instrument Selection

Run `tools/bod_refresh.py` each morning (before market open) to:

- Download & cache BOD instruments (`data/instruments_YYYYMMDD.json.gz`)
- Update `.env` → `UPX_BOD_PATH`
- Auto■select `UPX_INSTRUMENT_KEYS` (NIFTY/BANKNIFTY index + ATM CE/PE for nearest weekly expiry) using the current LTP

Post■refresh validation:

```
python upx_algo/tools/validate_post_refresh.py
```

- Confirms all selected keys exist in BOD data and have valid `lot_size` & `tick_size`
- Checks options fields (`option_type`, `strike_price`, `expiry`)
- Dry■runs lot/tick enforcement; outputs `data/validation_YYYYMMDD.json`

Scheduling (Option A/B/C)

A) Cron (Linux/macOS)

- File: `tools/schedule_bod.sh`
- Crontab:

```
50 8 * * 1-5 /bin/bash /absolute/path/to/upx_algo/tools/schedule_bod.sh
```

B) systemd (Linux)

- Files: `ops/systemd/upx-bod-refresh.service`, `ops/systemd/upx-bod-refresh.timer`
- Install:

```
sudo cp upx_algo/ops/systemd/upx-bod-refresh.* /etc/systemd/system/
sudo systemctl daemon-reload
sudo systemctl enable upx-bod-refresh.timer
sudo systemctl start upx-bod-refresh.timer
```

C) Windows Task Scheduler

- PowerShell wrapper: `tools/bod_refresh.ps1`
- Register task:

```
powershell -ExecutionPolicy Bypass -File .\upx_algo\tools\register_bod_task.ps1 -TaskName "UpxBodRe
```

- Or via `schtasks`: `tools/register_bod_task.bat`

GitHub Actions CI (Daily at 08:50 IST)

- Workflow: `.github/workflows/upx-ci.yml`
- Secrets required: `UPX_API_KEY`, `UPX_API_SECRET`, `UPX_REDIRECT_URI`, `UPX_ACCESS_TOKEN`
- Fallback sample instruments: `data/sample_instruments.json` (used if public URL blocks automation)

Notes & Tips

- **Proto compilation** is mandatory for Market Data V3 (binary Protobuf)
- Always rely on **BOD instruments JSON** (daily) and enforce `lot_size`/`tick_size` before placing orders
- Use **Portfolio Stream** to implement OCO (when SL or TP fills, cancel the other)
- Start with **PAPER**, then **LIVE (sandbox)**, and only then **LIVE (real)**

Next Steps

- Add instrument filters for your watchlist (EQ, F&O;, specific strikes)
- Extend ATM selection to pick multiple bands (ATM ± 1/2/3)
- Add reporting (PnL curve, drawdowns) for backtests
- Harden reconnection logic for WebSockets and add alerting

Document generated automatically from our setup conversation for quick reference.