🌟 Simple Guide: How to Use the IBKR Strategy Templates (ib_insync OR ibapi)

Works with the Read-Only Dashboard (auto-detects all logs)

This guide explains:

1. Folder structure
2. Copying the template
3. Editing the CONFIG section
4. Writing your strategy logic
5. Running the strategy (ib_insync or ibapi)
6. Viewing results on the dashboard

1️⃣ Folder Structure (Required)

Your project must look like this:

```
Quant_X_Dashboard_Monitor/
│
├── dashboard_read_only.py
├── .env
├── utils/
│   ├── client_id_manager.py
│   └── performance_metrics.py
│
└── strategies_runner/
    ├── asset_runner_insync.py      (optional)
    ├── asset_runner_ibapi.py       (optional)
    └── logs/
        ├── ASSET_RUNNER/
        │   └── trade_log.csv
```

├── IYW_GLD_volguard/

│ └── trade_log.csv

└── ...

The read-only dashboard automatically loads:

strategies_runner/logs/<STRATEGY_NAME>/trade_log.csv

2️⃣ Copy Your Preferred Template

You have **two templates** available:

✓ ib_insync template

✓ easier to code
✓ more reliable event handling
✓ recommended for beginners and most strategies

Save as:

strategies_runner/asset_runner_insync.py

✓ ibapi template

✓ lower-level, more control
✓ more verbose coding
✓ good for those comfortable with raw IBAPI

Save as:

strategies_runner/ asset _runner_ibapi.py

3️⃣ Edit the CONFIG Section (Important)

In both templates, find:

APP_NAME = "MY_STRATEGY_NAME"

SYMBOL = "AAPL"

SEC_TYPE = "STK"

EXCHANGE = "SMART"

CURRENCY = "USD"

Change to match your strategy:

Example:

APP_NAME = " SPY_0DTE "

SYMBOL = "IYW"

SEC_TYPE = "STK"

EXCHANGE = "ARCA"

CURRENCY = "USD"

This must match the folder where logs will be written:

strategies_runner/logs/SPY_0DTE/

4️⃣ Implement Your Strategy Logic

Both templates include this method:

```
def compute_signal(self, price: float) -> Optional[str]:
    return None
```

Replace with your logic:

Example SMA:

```
def compute_signal(self, price: float):
    self.prices.append(price)

    if len(self.prices) < 30:
        return None

    fast = np.mean(self.prices[-10:])
    slow = np.mean(self.prices[-30:])

    if fast > slow and self.current_position == "NONE":
        return "BUY"
```

```
    if fast < slow and self.current_position == "LONG":
        return "SELL"

    return None
```

✓ Both templates interpret "BUY" and "SELL"
✓ Both templates avoid looping trades
✓ Both templates enforce USD 2,000 cap & min 1 share

5️⃣ Run the Strategy

Option A — Run the ib_insync version

cd strategies_runner

python spy_0dte_insync.py

You will see something like:

[SPY_0DTE] Connecting to 127.0.0.1:7497...

[SPY_0DTE] Subscribed to live market data.

Option B — Run the ibapi version

cd strategies_runner

python spy_0dte_ibapi.py

You should see:

[SPY_0DTE] Connecting to IBKR...

[SPY_0DTE] Requesting market data...

6️⃣ Strategy Logs Are Written Automatically

Whenever a fill is confirmed by IBKR, both templates write:

strategies_runner/logs/SPY_0DTE/trade_log.csv

Each row contains (dashboard-compatible):

timestamp, symbol, action, price, quantity, pnl,

duration, position, status, ib_order_id, extra

✓ ib_insync → uses Trade.updateEvent
✓ ibapi → uses execDetails

Both templates log **only real fills**.

7️⃣ Open the Dashboard

Run:

streamlit run dashboard_read_only.py

The dashboard **auto-discovers** any strategy folder with:

trade_log.csv

You will see:

✓ Strategy name
✓ Trades table
✓ Equity curve
✓ Drawdown
✓ Metrics (CAGR, Sharpe, Profit Factor…)
✓ Per-strategy panels

8️⃣ Stopping a Strategy

Press:

CTRL + C

or close the terminal.

9️⃣ Running Multiple Strategies

You can run any combination of runners:

Example (ib_insync + ibapi mixed):

python spy_odte_runner_insync.py

python volguard_runner_ibapi.py

python ibm_fourier_runner_insync.py

python spy_0dte_runner_ibapi.py

Each:

- Gets a unique client ID from client_id_manager
- Writes to its own log folder
- Appears as its own row in the dashboard

✓ No interference
✓ No connection conflicts
✓ Full scalability