ChatGPT

XAUUSD Scalper Redesign – Comprehensive Analysis and Plan

1. Context of the Current System (v1)

EA Structure & Logic: The existing **XAUUSD-GOD v1** EA was originally intended as a scalper but behaves more like an infrequent "liquidity sweep" sniper. It runs on the **M15 timeframe** (after a last-minute change from M5) and includes numerous modules: - **Regime detection:** Uses ADX to classify market as trending or ranging [1]. If ADX ≥ 25, it labels **REGIME_TREND**; otherwise **REGIME_RANGE**. This single threshold decides which strategy to apply. - **Range strategy (Liquidity sweep):** In range mode, it scans for a tight consolidation and then a **false breakout** (price wicks beyond the range then closes back inside). The code `ScanAndSignal_LiquiditySweep()` identifies a 12-bar range of width 100–800 points and checks the last bar for a wick >20 points outside the range that closes back inside [2] [3]. If found, it generates a **market entry** signal in the opposite direction (expecting a mean reversion) with stop-loss ~1.2×ATR and take-profit ~2×SL distance [4] [5]. - **Trend strategy (Breakout):** In trend mode, it looks for a **20-bar breakout** beyond recent highs/lows in the direction of a long-term trend. It requires price above a 200 EMA for long (or below for short) [6] [7] and the last bar to close ≥3 points beyond the 20-bar high/low [8] [9]. If so, it issues a **pending stop-order** entry 5 points beyond that breakout level (to confirm momentum) [10] [11], with stop-loss ~1.5×ATR and TP ~2×SL distance. - **Risk management:** A fixed 2% risk per trade (or fixed lot) is used. Lot size is computed so that if SL hits, ~2% equity is lost [12] [13]. The EA ensures volume within min/max limits and normalizes lot to broker step size [14] [15]. There is a rudimentary **daily drawdown check**: trading stops if current equity drawdown exceeds 5% from the day's peak [16]. - **Trade management:** Once in a trade, a separate manager will apply **break-even moves and trailing stops** if enabled. In v1's inputs, trailing was enabled (ATR-based trailing stop) but partial take-profits were disabled by default [17] [18]. The EA calls `TM_ManageOpenPositions()` on every tick to adjust stops: it moves SL to entry after +150 points profit, trails by 1×ATR or by fixed steps depending on mode, and could close a partial at a set point if that were enabled [19] [20]. - **Entry gating:** Before seeking any signal, the EA enforces several filters: - It only evaluates signals on a **new bar** (one trade decision per bar). In v1, this was actually a new M15 bar [21], which significantly slowed trading (only 4 bars/hour vs 12 on M5). - It checks a `CanTradeNow()` gate requiring spread <= Max_Spread and ATR >= Min_ATR [22] [23]. In v1, Max_Spread_Points was set extremely high (1500 points) effectively never blocking trades, while ATR filter required ATR≥100 points (to avoid ultra-low volatility periods) [24]. - It ensures trading is allowed on the given weekday (Mon–Fri) [25] and that the daily drawdown gate (5%) hasn't been hit [26]. - **Logging and safety checks:** The EA logs events for debugging (e.g., whether gates blocked a trade due to spread or drawdown [27]). It also checks that SL is not zero before sending orders [28] and uses a max slippage setting of 50 points on execution [29].

Why v1 Fails as a Scalper: In practice, this design produced **very low trade frequency** (~20–30 trades per year). The overly strict filters and multi-condition requirements mean only "picture-perfect" textbook scenarios trigger trades. Specific issues include:

- **Timeframe Mismatch:** The logic was implicitly designed for M5 (fast scalping of small moves), but the EA was run on M15 [30]. This triples the duration of setup patterns, making signals far less frequent. For example, a 12-bar range on M15 spans 3 hours (versus 1 hour on M5), so the **liquidity sweep** setup rarely occurs. Similarly, waiting for a 20-bar breakout on M15 (~5 hours of data) misses many shorter intraday breakout opportunities. The slower timeframe also means stops (based on ATR) and profit targets were larger in absolute terms, reducing the risk/reward of quick scalps.
- **Over-Filtering:** The EA requires multiple conditions to line up perfectly. In range mode, it needs a well-defined consolidation of a certain size **and** a single-bar false-breakout **and** an RSI confirmation (last bar's RSI <35 for longs or >70 for shorts) [31]. Many potential mean-reversion scalps are skipped because they didn't check every box. In trend mode, price must break a multi-hour high/low by a tiny margin (3 pts) *and* close beyond it in the direction of the long-term trend. If any piece is missing (e.g. price spikes through but also closes beyond by too much, or doesn't close beyond enough), no trade. Such narrow trigger criteria drastically reduce frequency.
- **Session & News Ignored:** The system doesn't explicitly account for trading sessions or news. There is a session input (7–18 GMT) in v1, but it was not actually enforced in code (no time-of-day check in `CanTradeNow()` or elsewhere). This means the EA might avoid some volatile periods by chance but not consistently. It could even trigger a rare trade during off-hours if all other conditions align. By not focusing on the high-liquidity **London/New York hours** when scalping is most effective, it further lowered its trade count.
- **Single-Strategy Focus:** The EA essentially had two modes (range reversal or trend breakout), each of which is very selective. In a strongly trending market, the range mode never triggers; in choppy markets, the breakout mode rarely triggers. Because these modes don't overlap or adapt dynamically, the system spends a lot of time "idle." It does not exploit more frequent scalping opportunities like minor pullbacks or momentum bursts – it only trades extreme scenarios (range fake-outs or major breakouts).
- **New Bar Restriction:** Only acting on new M15 bars means intra-bar signals are ignored. Fast transient moves that revert within the same candle are missed entirely. Scalpers often need to react **within** a bar (especially on M5 or lower) or on smaller timeframe cues; v1's design misses those chances by construction, further contributing to the "sniper" behavior.
- **Excessive Safety Margins:** Some parameters in v1 were chosen very conservatively, effectively filtering out many potential trades. For instance, requiring ATR $\geq$ 100 points (i.e. $1.00) ensures a minimum volatility, but in calmer conditions the EA simply won't trade at all. Likewise, the range width had to be at least 100 points – smaller micro-ranges (e.g. 50-point range on M5 that could still yield a scalp) were ignored. These conservative thresholds contribute to inactivity. While they aim to avoid false signals, they ended up excluding a large portion of viable scalping setups.
- **No Adaptation:** The EA did not adjust its behavior if it was missing too many trades. It lacked any self-tuning or multiple setups to increase frequency. It rigidly waited for *perfect* patterns that rarely materialized on gold. Real market price action is often messier than these ideal patterns, so the EA sat on the sidelines most of the time.

In short, v1 fails as a scalper because it trades **too infrequently** and inflexibly. Its strict regime split and multi-condition filters produced a "rare sniper" system rather than a high-activity scalper. Gold's intraday behavior often didn't meet the EA's narrow criteria. Combined with running on M15 and ignoring many quick opportunities, the EA could not generate the 100+ trades/year expected of a scalping strategy.

Instead it only caught occasional “picture-perfect” moves, missing out on the myriad smaller moves a scalper should capture. The result is an under-trading system with long dry spells – the opposite of an agile scalper.

(No v1 trades are taken during typical days; it only triggers on exceptional candles, hence the low count.)

2. Research – Real-World XAUUSD M5 Scalping Benchmarks

To redesign the strategy, it’s crucial to understand how **real gold scalping strategies** behave in terms of frequency, risk, and execution. We investigated high-credibility sources including active trader forums, verified track records, and institutional insights. Below we summarize key quantitative benchmarks with citations:

- **Trade Frequency:** Scalping is generally characterized by **frequent trades**. Professional manual scalpers might take **20–30 trades per day**, holding each for only a couple of minutes. For example, one source notes a scalper using 5-second charts could make *“twenty or thirty trades per day, with each trade being active for only two minutes.”*. By contrast, more moderated strategies (especially automated EA scalpers on M5) tend to trade a few times per day. Some commercial gold scalping EAs explicitly limit trades – e.g., one EA opens **2 trades per day** (one buy, one sell) consistently. Another live strategy (“XAUUSD Scalper Plan B”) executed **839 trades over 10 months**, roughly 20–25 trades per week. **Contradiction:** Sources therefore range from high-frequency human scalpers (dozens of trades daily) to EA-based approaches doing only a couple a day. Our target of **100–300 trades/year** (~2–6 trades/week) is on the lower end for “scalping.” It prioritizes quality over quantity, whereas many scalpers aim for numerous quick wins. We must be mindful that too few trades (like v1’s ~20/year) is ineffective, but too many can incur costs and noise. **Conclusion:** Aiming for around **0.5–1 trade per day on average** is a reasonable middle-ground for an automated gold scalper, given that some days will have multiple opportunities and others none. This is far below the manual scalping extremes but much higher than v1’s sniper rate.

- **Typical Stop Loss (SL) and Take Profit (TP):** Gold’s volatility means scalpers often use modest pip targets with tight stops, or ATR-based dynamic sizing. One popular approach is aiming for **about 10 pips ($1.00) profit per trade** on gold in active sessions [32]. For instance, a Forex Factory strategy “Trade Gold for +10 pips Daily” notes gold often moves 20–30 pips in London morning, so the goal is to **“scalp only 10 pips out of it.”** [32]. They use a **fixed 10 pip partial and 20 pip full TP**, with stop-loss just beyond the trigger candle (often ~10–15 pips). This implies risk:reward around 1:1 to 1:2 on each trade. Many scalping systems indeed target **1:1 to 1.5:1 RR** per trade. For example, a Reddit user mentions **1R–1.5R scalps** as a common goal in their strategy. Our own design options reflect this: *Option A* proposes TP = 1.5×SL (i.e. 1.5R) and *Option B* effectively targets ~2R (with partials along the way). ATR-based stops are also prevalent – several EAs compute SL as a multiple of ATR (commonly 1.0× to 2.0× ATR for scalping). The v1 used ATR*1.2–1.5, which aligns with industry practice. One signals provider explicitly recommends ATR-based stop sizing for gold scalping, as it adjusts to volatility. Benchmark: We’ll design for roughly 10–30 pip stops (depending on volatility) and 15–50 pip profits**, corresponding to 1–2R outcomes. These small targets match gold’s intraday ranges – e.g. gold’s average M5 ATR during London open is ~20–30 pips [32], so a 10–15 pip scalp is a realistic bite of that move.

- **Holding Time Distribution:** True scalping trades are very short in duration. Professional scalpers often close positions **within minutes** – the example above was ~2 minutes on average. For EAs, we see slightly longer but still intraday holds. A Myfxbook analysis of one gold scalper EA shows the **average trade length was a few hours at most, with the majority closed in under 1 hour** (implied by the high trade count and intraday focus). Many scalping strategies enforce a **time-based exit** to avoid overnight risk – for instance, closing any open trade by end of session or if it exceeds a certain duration (common values are 1–3 hours max). This prevents getting stuck in unexpected swings. **Benchmark:** We will cap the **max holding time at 2–3 hours** per trade, and likely most trades will close well before that (many within 5–60 minutes as targeted). This is consistent with our target of quick scalps and is supported by best practices – e.g., one source explicitly advises not to hold gold scalping trades beyond a couple of hours, given gold's tendency for sharp reversals if a move hasn't materialized.

- **Preferred Trading Sessions:** There is consensus that **London and New York sessions** are prime time for XAUUSD scalping. Gold typically has high liquidity and volatility in the London morning through the NY afternoon. One strategy notes: *"Gold (XAUUSD) easily has an ATR of 20-30 pips during London open"* [32] , making it ideal for grabbing quick profits. Conversely, the Asian session is quieter for gold; spreads widen and moves are smaller, making scalping harder. A signals provider quantifies typical conditions: during **Quiet Asia**, gold spreads are ~15–25 points (0.15–0.25) with minimal slippage, whereas at **London open (08:00–09:30 GMT)** spreads run ~20–50 points and **NY overlap** ~25–60 points, with much more momentum. Despite slightly higher spreads, the volatility in London/ NY provides better trade opportunities. Many experienced traders avoid the first few minutes of session opens (due to spread spikes) but trade the bulk of those sessions. Our design will **strictly trade London and New York hours only**, avoiding the Asian session altogether. Specifically, we'll allow entries roughly between **07:00 and 18:00 GMT** (covering London open through the first half of New York). This aligns with common practice as seen in v1's intended session inputs (London open to NY close) and forum strategies focusing on London breakout setups. Additionally, we will implement a **Friday cutoff** – many gold scalpers close positions by Friday 17:00–21:00 GMT to avoid weekend gaps. We'll flat all trades by Friday 21:00 GMT as required.

- **Indicators & Entry Triggers:** Successful gold scalping strategies often use a combination of a **trend filter** and a **short-term trigger**:

- **Trend filters:** Many traders rely on a higher timeframe trend or a moving average to bias trade direction. For example, using a **200 EMA** on M5 or referencing H1 trend is common (v1 did this by requiring bias long if price > EMA200 [7]). At least three sources (our v1, the example FF strategy, and multiple MQL5 EAs) emphasize aligning scalps with the larger trend to improve win rate.
- **Momentum/Volatility triggers:** Tools like **RSI** or **Stochastics** are frequently used to time entries – either entering on **oversold/overbought extremes** in the direction of the trend (a quick mean reversion within trend), or on momentum breakouts. In our research, **RSI** appears repeatedly: v1 used RSI<35 or >70 as a confirmation for reversals, and other EA descriptions also list RSI among core indicators. Bollinger Bands and ATR-based bands are also used by scalpers (mentioned in code and strategies on forums) to gauge volatility contractions before a breakout or to identify overextension.
- **Price action patterns:** Many strategies require a simple candle pattern to confirm entry – e.g., an **engulfing candle** or a break of a minor support/resistance. Tiptoptrade's 10-pip strategy, for instance, looks for a big candle out of Asia range followed by a smaller pullback candle, then an

engulfing in the original direction [33]. This is essentially price action confirming the momentum or reversal. Several sources stress that **scalping is primarily technical** (you rely on charts, not fundamentals), and fast patterns on M1–M15 charts trigger the trades.

- **Volume and volatility filters:** Some advanced scalpers include a volume spike filter (like v1 did for breakouts) or an ADX threshold to confirm strong movement. For example, one breakout example noted **volume tripling the 20-period average** to validate the breakout. Another EA requires a minimum volume or volatility condition (e.g., don't trade if ATR is below X, which v1 also had). At least three unrelated sources (v1, a signals blog, and QuantVPS guidance) highlight ATR or volatility-based gating as important to avoid dead markets. We will incorporate **ATR and possibly ADX** in filtering – ensuring enough volatility is present and possibly adjusting strategy if volatility regime changes.

Summary of Common Indicators: EMA(50 or 200) for trend appeared in multiple systems (for trend bias) [7], RSI(14) appears across different strategies for momentum/reversion signals [31], and ATR is used for stop sizing and volatility gating by many. We will prioritize these in our design since they have broad support. Notably, **no single magical indicator** was universal – successful scalping comes from a combination that suits the strategy's logic (e.g., trend + pullback signal). We'll use a **max of 2–3 indicators** in combination to keep the logic simple and robust, as per our anti-overfitting mandate.

- **Spread Tolerance & Execution Assumptions:** Gold's spreads can vary widely by broker and session. On a **good ECN account**, typical spreads during active hours are around **20–30 points (0.20–0.30)**, and can widen beyond 50 points in volatile moments. Market makers may quote a fixed spread (e.g., 30–50 points) but could slip internally. For scalping, **low spread and low slippage** are critical. Institutional references indicate XAUUSD's target spread is ~41 points on major brokers (about $0.41) – this matches our assumption that 50 points ($0.50) is an absolute max to even consider trading. Indeed, our broker constraints will enforce not trading if spread > 50 points. During major news (NFP, Fed decisions), spreads can **balloon to 70–150+ points** and slippage becomes highly likely. We must avoid trading in those times (hence a news filter).

Execution speed is also paramount: scalping profits can be erased by a few points of slippage. A VPS located near the broker's server is strongly recommended. One source specifically notes that **ultra-low latency (0–1 ms)** execution can improve gold scalping outcomes and reduce slippage. It states *"for strategies like scalping or breakout trading, where speed is everything, dependable infrastructure can make all the difference."*. Additionally, using limit orders when possible can avoid negative slippage (at the cost of maybe missing some trades) [34]. In practice, many scalping EAs use market orders for quick entry but accept some slippage. Our design will assume: - **Raw ECN spreads** (with commissions) to keep spreads ~15–30 points on average. - **Max slippage tolerance ~50 points** – we won't accept more than $0.50 slippage; the EA will attempt to set that (v1 already had Max_Slippage_Points=50). - **5ms or lower latency** to broker – the user must run this on a high-speed VPS (e.g., in LD4 or NY4 if broker is there). If execution delays >500ms are detected, the EA should halt new entries temporarily (per constraint). - **No trading during illiquid periods:** around daily roll-over, or moments of extreme spreads. For instance, just after 17:00 NY (when spreads spike), or immediately when a big news hits – we will build in a delay.

These assumptions mirror what experienced traders expect. One education piece enumerates: *Quiet Asia spread ~15–25 pts, London open 20–50 pts, NY overlap 25–60 pts*. It also warns that slippage, while minimal in calm periods, is "common on first impulse" of NY open or big data. Therefore, our EA will include **spread filtering** and likely skip the first minute of major session opens or news releases to let spreads normalize [35]. We'll also implement the rule to pause if execution is lagging or if slippage gets too high unexpectedly.

In summary, real-world data suggests our scalper should operate **only in the high-liquidity sessions (London/NY)**, target **small quick profits** with **stops on the order of 10–20 pips** (adjusted by ATR), and ensure robust risk controls. Spreads must be kept under control (ideally <30 points) and the EA must **explicitly avoid known danger periods** (major news spikes, illiquid hours) to survive. Many strategies that scalp gold successfully have **win rates in the 70–85% range** with careful risk management, leading to moderate but consistent profits [36]. This informs us that **frequent small wins** are the way to go, but we must also handle the occasional large move against us (via hard stops and loss limits). All these insights from active traders, market stats, and institutional knowledge will be baked into the v2 strategy design.

(Sources: ForexFactory threads, MQL5 EA descriptions, Myfxbook stats, FX broker data, and VPS providers as cited above.)

3. Broker & Execution Assumptions (Mandatory Constraints)

To ensure the new scalper performs in real conditions, we impose strict **broker and execution constraints** that the strategy must operate within:

- **Account Type:** Only use an **ECN or RAW spread** brokerage account. This gives the tightest spreads on XAUUSD with a commission, as opposed to standard accounts with marked-up spreads. The EA will assume spreads are usually in the 15–30 point range (0.15–0.30 pips) during normal trading, with an absolute **hard cap of 50 points**. If the spread exceeds 50 points at any moment, the EA will not open new trades. (In v1, `Max_Spread_Points` will be adjusted down to 50 from the old 1500 to enforce this.)

- **Spread Filter:** As part of pre-trade checks, the EA will call `SpreadOK()` and require `spread <= 50` points to proceed [37] [22]. We will also incorporate a dynamic element: if spread is above typical (e.g. >30 points) but below 50, the system may decide to skip **non-critical trades** or reduce position size. This prevents entering full-risk scalps in suboptimal conditions. Empirically, spread tends to widen beyond 30 points in off-hours or exactly at big news releases, so this filter keeps us out of trouble.

- **Slippage Control:** The EA will set a **Maximum Slippage** of 50 points on all orders (as in v1, using `g_trade.SetDeviationInPoints(50)` [38]). Any fill that would slip more than this will likely be rejected. Additionally, if we detect that actual fills are consistently slipping (e.g., if `GetLastError()` returns a slippage error or if trade entry price differs from requested by >50 pts), the EA could take adaptive action: such as pausing trading for a while or switching to limit orders. (The user should also monitor the broker execution quality, as excessive slippage means the broker or network is not suited for scalping.)

- **Latency & Execution Speed:** It is **mandatory to run the EA on a VPS** with <5ms latency to the broker's trade server. If the EA measures order execution time > 500 ms (half a second), it will log a warning and **temporarily halt new entries**. A delay that large could indicate server issues or an overloaded connection, which can be deadly for scalping. The EA can implement this by timestamping when it sends an order and comparing to fill time; or simpler, by monitoring if ticks have come but orders remain unfilled for >500 ms. On detection, it will not initiate further trades for a cooldown period (perhaps a few minutes) and will advise via log to check the VPS/connection. As research noted, top VPS providers offer 0–1 ms execution which is ideal – we expect the user to aim for that environment.

- **Tick Data and Realistic Backtests:** All backtesting and optimizations for v2 must be done with **100% tick data**, variable spreads, and a simulation of execution delay. We will not rely on M15 OHLC testing as was possible in v1. The scalper's edge comes from fine movements and real spreads, so testing must replicate that. The strategy tester should incorporate a ~100–200ms trade execution delay (MetaTrader can simulate this) and variable spread data if available. This ensures the EA's performance in testing is closer to live reality (no overestimation due to assuming zero slippage or fixed tiny spreads).

- **Spread and Slippage Adaptation:** The strategy explicitly incorporates rules to handle sudden changes:

- **Spread Spike Pause:** If a sudden **spread spike** occurs (e.g., spread triples within <60 seconds), the EA will assume market liquidity dropped (perhaps news or rollover) and **pause entries for 10 minutes**. For example, if average spread was 20 points and it jumps to 60+, we halt new trades. This can be implemented by tracking a moving average of spread each minute and comparing. The rationale is that in such moments, even if our signals trigger, execution cost is too high and unpredictable.
- **Flash Crash Protection:** If an abnormal price spike or "flash crash" is detected (for instance, an M1 candle of >500 pips ($5) range), the EA will assume extremely unstable conditions and **halt trading for 30 minutes**. This implements the "flash crash rule": essentially, if gold moves hundreds of pips in a minute, we step aside to avoid getting whipsawed. The EA can detect this by monitoring the high-low range of each new 1-minute bar (via `iHigh/iLow` or tick data). If range > 500 pips, set a flag to block new trades until 30 min passes.

- Both the above situations will be logged (so the user knows why no trades are happening) and are in place to protect against extreme volatility events that can slip stops far beyond expected.

- **One Instrument Focus:** The EA will trade **XAUUSD only** (as specified). If the user also runs a similar EA on XAGUSD (silver) or other instruments, we ensure no conflict. Specifically, we state that if both gold and silver give a signal at the same time, only one should trade (gold preferred). In practice, since our EA is for XAUUSD, it simply won't be looking at XAGUSD at all. But in case of a multi-symbol extension, we would implement a mutual exclusivity such as a global flag or magic number scheme to avoid simultaneous metals positions. For now, the scope is single-instrument: focusing all risk on gold where we have the edge.

- **No Hedging or Multiple Positions:** We will allow **only one open position at a time on XAUUSD**. This simplifies execution and risk. If a signal arises while a trade is already open, it will be ignored. Many brokers (especially US) don't allow hedging, and even if hedging is allowed, opening opposite trades is not in our plan. The EA will use a dedicated magic number for its trades (as v1 did with `MAGIC_BASE + Magic_Offset` [38]) to track its position. Before opening a new trade, it will check if there's already an open position with its magic; if yes, it skips entry until that is closed. This ensures we don't stack multiple scalps in unpredictable ways.

- **Order Types:** The v2 strategy will primarily use **market orders** for entry, given the need for quick execution (Option A in trade management involves straightforward entries/exits). In v1, pending orders were used for breakouts. We may still occasionally use stop orders for breakouts if the new logic calls for it (for example, if we implement a breakout setup, we might place a buy stop a few pips above a level). However, market orders give more control in fast moves. We will utilize **stop-limit** orders for news avoidance or breakout protection if needed – e.g., if we do trade a breakout, we can use an MT5 `BuyStopLimit` to cap slippage. But by default, market orders with slippage caps should suffice. The EA will ensure to always set a Stop Loss with every order (no open trades without SL, to cap risk – essential in volatile gold). It will also set a Take Profit at the intended target (especially if we might not be able to monitor continuously, a TP ensures profit is taken).

In essence, these constraints mean the strategy will only deploy in a broker environment conducive to scalping (tight spreads, fast execution). It will actively avoid conditions where those assumptions break down (widened spreads, high slippage, extreme volatility). The **trading code will incorporate multiple safeguards** to enforce this: spread checks every tick, slippage limits on orders, time-of-day filters, and self-imposed trading halts during abnormal market states. By doing so, we significantly improve the chances that live performance will match our expectations and not be derailed by execution issues.

(In summary, if the broker's conditions deviate from our assumptions – e.g. if typical spread is 100 points or trades consistently slip – the EA will either not trade or the user should change to a more suitable broker. These assumptions are non-negotiable for the scalper's success.)

4. Target Performance for the New Strategy (v2)

The redesigned **v2 strategy** is aimed to meet specific performance and risk targets, which define success for this scalper. These targets are:

- **Symbol & Timeframe:** XAUUSD (Gold) on the **M5 chart**. All logic will be optimized for 5-minute candles, which balances opportunity and noise for scalping. (We expect roughly 12 bars/hour, 288 bars/day on M5 – plenty of granularity for intra-session trades.)

- **Trade Frequency:** Approximately **100–300 trades per year**, i.e. an average of 2–6 trades per week. This is a dramatic increase from v1's ~20/year, yet still restrained (we're not doing dozens per day). It acknowledges that high-probability setups in gold might occur a few times per week. The wide range (100 to 300) allows for market-dependent fluctuation – some volatile years nearer 300, quieter ones around 100. This frequency means the EA will be engaging the market regularly, giving enough sample trades to statistically realize its edge, without overtrading. (For reference, a mid-range target of ~200 trades/year is roughly 4 trades/week, which seems reasonable given gold's active days.)

- **Holding Time: 5–60 minutes average trade duration**, with an absolute **maximum of 2–3 hours**. The strategy is intraday only. Most winners should hit TP or exit within an hour; many quick scalps might close in under 15 minutes. Losing trades will hit SL usually within minutes to an hour as well. No trade will be left open beyond 3 hours – a hard time-based exit will close it if it hasn't hit TP/SL by then (since an unfulfilled scalp after 3 hours likely isn't going to succeed). This ensures we don't carry risk through session transitions or into low liquidity times. It also fits the definition of scalping/day trading – no overnight holds.

- **Per-Trade Risk: 0.25% – 1.0% of equity** at risk per trade (user-adjustable). By default, we plan to risk **0.5%** of account equity on each trade (half of one percent). This is lower than v1's 2% risk, aligning with a conservative scalping approach that might have many trades. We allow the user to choose up to 1% if they want more aggression, or as low as 0.25% if they want ultra-safe. Keeping risk small per trade ensures that even a streak of losses doesn't cripple the account (e.g., 10 consecutive 0.5% losses = 5% drawdown). Position sizing will be dynamic using the account equity at trade time to compute lots for the given SL distance. This also satisfies prop firm style risk limits, where low per-trade risk is essential.

- **Drawdown Target: Max peak-to-valley drawdown of 5–15%** of equity. We aim for the strategy to stay in single-digit drawdowns most of the time, with an allowable worst-case up to 15%. This is an aggressive goal given gold's volatility, but achievable by strict risk management and loss limits. It means the strategy should endure losing streaks without significant capital impairment. For example, if risking 0.5% per trade, a 15% DD implies around 30 consecutive losses (which is extremely unlikely if the strategy has an edge). Empirical reference: a well-tuned gold scalper often advertises <20% DD with high win-rate [36], so we're aiming even tighter at max 15%. The EA will incorporate measures (daily/weekly loss stops, etc.) to enforce this target (see Risk Profile section).

- **Profit Factor & Returns:** While not explicitly asked, hitting the above risk/drawdown targets should allow decent returns. If we assume ~50% win rate with 1.5R trades, profit factor ~1.2–1.5, we could see ~10%–30% annual returns at 0.5% risk per trade, with 10% DD (just ballpark). The user's mention of 20-40% monthly in a forum thread was deemed unrealistic without high risk, confirming our conservative stance. So we target steady growth with controlled risk, rather than huge ROI with huge DD.

- **Trading Sessions: London and New York sessions only.** Concretely, we will trade roughly from **07:00 GMT to 17:00 GMT** Monday–Friday:

- Start ~07:00 covers London open (08:00 London local time).
- End ~17:00 covers the bulk of New York session (12:00 New York local, which is a couple hours before NY close when liquidity starts dropping). We might extend to 18:00 GMT if good opportunities still arise, but certainly no new trades after that. The input will allow some adjustment, but default is 7–18.
- Absolutely **no trading during the Asian session** (approximately 22:00–06:00 GMT). The EA's session filter will block entries in that window. Asian hours often see gold range in tight bands – not worth scalping given higher spreads then.

- **Friday rule:** Additionally, as mandated, we will close all trades by **Friday 21:00 GMT** (a bit before the market weekly close). And we will stop taking new trades well before that (likely after Friday 17:00 or as configured). This avoids weekend gap risk. So effectively, Friday trading might be cut short – maybe only London session on Friday and early NY, then shut down.

- **Resilience to Market Conditions:** The strategy should withstand **news spikes and volatility shocks** without blowing up. This means:

- If a sudden news spike happens while we have a trade, our stop-loss is in place (and possibly a slippage control). We accept the loss and move on – position size is small enough to handle it (0.5% risk).
- We avoid entering right before known major news (NFP, CPI, Fed etc.), so ideally we're not in the market during the most violent moves. The EA will include a news filter (see Edge Cases) to skip those times.
- If volatility structure changes (e.g., gold ATR doubles for some weeks due to crises), our ATR-based stop sizing will adapt, and our loss limits will ensure we perhaps trade less or with smaller size during that period to keep drawdown in check.

- Surviving a "volatility shock" means even if gold makes an abnormal move, our protective rules (flash crash halt, etc.) minimize damage. We won't be adding to losing positions or doing anything that could exacerbate losses.

- **No Sniping Behavior:** Importantly, the v2 logic is designed **NOT to revert to sniper-style trading**. That means it will take more *bread-and-butter trades*, not just extraordinary ones. With 100–300 trades/year, we ensure it's engaging with the market regularly. It will capture many small moves (some mean reversion bounces, some momentum bursts) rather than only the rare perfect setups. So performance will come from accumulating many modest wins rather than a few jackpots. This also smooths the equity curve, which is desirable.

By meeting these targets, the new EA should produce a steady trading performance akin to what seasoned scalpers achieve: - **Drawdowns limited to mid-teens%** even in stress periods, - A stream of **short trades primarily in active sessions**, - **Risk per trade low** enough to survive losing streaks, - **Consistent enforcement of stops and risk limits** to guard against blowouts.

If we find in forward testing that the trade frequency or drawdown deviates, we'll adjust parameters accordingly (e.g., loosen/tighten entry criteria to hit the trade count, or reduce risk if DD edges up). But these targets are the guiding stars for v2's design and will be encoded into the strategy's rules.

(The end goal is a strategy that a prop firm or risk-conscious trader would be comfortable running: frequent small gains, strict risk management, and no surprise large losses.)

5. Strategy Design Requirements

Designing v2 involves balancing higher frequency with robust filtering to avoid random noise trades. We impose formal **design constraints to prevent overfitting** (per 5.1), specify trade management approach (5.2), outline edge-case handling (5.3), and define the risk management profile (5.4). Each aspect is detailed below:

5.1 Anti-Overfitting Constraints (MANDATORY)

To keep the strategy simple, generalizable, and not overly curve-fit, we will limit the complexity of entry logic:

- **Maximum 4 Entry Conditions (ANDed):** The core entry setup will consist of at most **4 conditions combined with AND logic**. This does not count trivial checks like "trade only during sessions" or "spread < X" – those are gating, not part of the pattern. The 4 conditions refer to the technical signals that must align for an entry. Keeping conditions ≤4 ensures the strategy isn't a convoluted checklist that rarely triggers. It forces us to choose the most impactful factors. For example, our entry might require: (1) Trend bias is bullish, (2) Price dips to an oversold level, (3) A bullish candle pattern appears, (4) ATR above minimum. That's 4 conditions. We will not add more like "and RSI divergence and pivot level match" etc., as that would overfit. Each condition we include must have a strong rationale and benefit.

- **Maximum 2 Exit Conditions:** Exits will be kept simple as well. Ideally, we will use **one primary exit condition** (hit the TP or SL) and possibly **one secondary** (e.g., time-based exit or manual close if an opposite signal appears). We will *not* have a laundry list of exit rules (no "if RSI crosses X, or if MAs cross, or if volatility drops, then exit" – too many exit rules cause whipsaw). For v2.0, the exit model is straightforward (fixed SL/TP or trailing as chosen). So in practice our exit conditions are: (1) SL hit, (2) TP hit, and optionally (3) max time reached – that's it. This adheres to the constraint (time exit is a failsafe, not an extra "condition" in the sense of indicators, but we count it as one of the two since it can trigger an exit). By capping exit criteria, we avoid micro-managing trades in a way that was optimized on historical quirks.

- **Filters >20% Trades = Optional:** For any filter that **eliminates more than 20% of potential trades**, we must make it *optional* (or seriously reconsider it). This is a critical anti-overfitting rule. It means if we introduce a condition that cuts out a large portion of trades, we suspect it might be too restrictive unless strongly justified by false-signal reduction. For instance, if we add an ADX filter that blocks trades in low volatility and it ends up skipping 30% of trades, we must evaluate if those skipped trades were mostly bad (if yes, fine; if they include many good trades, that filter may be overfit). In practice, we will implement most filters with an enable/disable input so we can test the difference. For example, we might include a **volume spike filter** or an **ADX threshold** – but if disabling it increases trades by 30% with only slightly worse performance, we might leave it optional or off by default. Every filter's impact on trade count and win rate will be measured. The documentation will note which filters are optional. The session filter and risk filters are not optional (they're structural), but technical filters like "require RSI confirmation" can be optional if too restrictive (v1's RSI confirm cut out some trades – in v2 we might allow disabling that if it filters >20% trades).

- **Justification for Each Condition:** We will explicitly justify each condition with **quantitative reasoning**. That is, for each entry condition or filter, we'll state how it contributes: "In backtests, adding Condition X increased trade frequency by Y% while *reducing false signals* by Z%." For example, *Trend Filter:* "Allowing only trend-aligned trades cuts about 50% of trades (all counter-trend ones) but improves win rate significantly, reducing false signals by more than 50%. Worth keeping." Or *Session Filter:* "Eliminates ~30% of day (Asian hours) which contributed only ~10% of profitable trades, thus it filters low-quality trades effectively." We will compile these stats during optimization. For now, the expected rationale is:

- **Trend filter (EMA or H1 bias):** May remove ~50% of potential trades (no counter-trend entries), but expected to *increase win rate by a large margin* (perhaps turning a 45% win rate strategy into 60%+ by avoiding going against the tide). Many false breakouts are avoided by not trading counter-trend.
- **Volatility filter (ATR > threshold):** Might remove ~10–20% of trades that occur in very low volatility times. Those trades tend to stall or hit stop due to lack of movement, so filtering them should improve the win rate or profit factor. For instance, we'll justify "requires ATR(14) on M5 > 0.1 (i.e. >10

pips) – this cuts ~15% of trades that occurred in flat markets, and about 80% of those trades were losses due to spread costs and random noise. So this filter reduces false signals significantly."

- **Oscillator confirmation (RSI or similar):** This might filter, say, 20–25% of signals that didn't reach an extreme condition. We'll have to check if those filtered signals were actually bad trades. If yes (e.g., requiring RSI < 30 for buy filtered out many weak pullbacks that often kept dropping), then we justify it like "it reduced trades 20% but those were low-quality, raising overall profitability". If it filters too many good trades, we'll adjust threshold or make it optional.
- **Candle pattern confirmation:** This will slightly reduce trades (maybe ~10% reduction if requiring an extra confirming candle) but ideally weeds out head-fakes. We might show that "without requiring an engulfing candle, the strategy took X trades with Y% win; with the requirement, it took slightly fewer trades but win rate improved by Z% – indicating it filtered out losers."

The overarching principle is to **keep the strategy as simple as possible** while still effective. We want broad strokes conditions that are explainable and rooted in market logic (trend, momentum, volatility). By limiting conditions and making heavy filters optional, we ensure the EA remains adaptable if market behavior changes. If an optional filter becomes counterproductive in the future, the user can disable it. We will clearly label those in inputs.

These anti-overfitting measures will make the strategy easier to maintain and less likely to fail going forward. Simpler strategies often generalize better – we're deliberately avoiding a complex web of rules that fit only past data.

5.2 Trade Management Rules

We will implement **one main exit model** in v2.0, and plan for an alternative in v2.1. Here are the options and the chosen approach:

Option A (Recommended for v2.0 – Simple & Robust): - **Full Take-Profit at 1.5R:** The entire position is closed when profit reaches 1.5 times the risk (i.e., if SL was 100 pips away, TP is 150 pips in profit). This is a fixed Risk-Reward exit. It locks in a decent profit multiple while maintaining a >50% win probability in a good strategy. - **Full Stop-Loss at 1R:** If price hits the stop level (1R = risk per trade, e.g., -100 pips in the example), the entire position is closed for a full loss. No partial stops, no widening – it's a hard stop. - **No Partial Close, No Trailing:** There will be no profit taken before 1.5R and no dynamic trailing of the stop. The trade either hits +1.5R or -1R (or perhaps exits at time limit). This is the simplest management approach.

Justification: This model is straightforward to implement and avoids the pitfalls of trailing stops in a choppy instrument like gold. Trailing too early often gets you stopped out at breakeven or small profit, missing the full move. Partial closes can dilute profits if the remaining portion doesn't run far. By taking the full position off at a moderate target, we **ensure winners bank a good profit**, and losers are cut at a predefined small loss. This also simplifies analysis – each trade is either +1.5R or -1R (with a few might be +0 if time exit triggers, but mostly binary outcomes). Many scalping approaches thrive with ~1.5R targets: you need roughly 40% win rate to break even at 1.5R, which is achievable. If our strategy can do, say, 55–60% wins, this RR yields solid profitability.

Additionally, not trailing stops means we don't get taken out by normal retracements in a still-valid move. Gold often pulls back before hitting a further target; a trailing stop might prematurely cut off a trade that would have been a winner at 1.5R. By avoiding that, Option A lets the trade idea fully play out. Simplicity also reduces execution overhead (no need to constantly modify SL) and risk of error. Therefore, we recommend Option A as the baseline for v2.0. It's easier to optimize fewer parameters (just one TP level and one SL level ratio, which we've set at 1.5:1 by rationale and can fine-tune slightly).

Option B (For future v2.1 – Advanced management): - **50% Partial Close at 1R:** When price reaches +1R (i.e., it has moved in profit equal to the initial risk), we immediately close half of the position, securing a profit of +0.5R on that portion. - **Move SL to Breakeven:** At the same point (1R profit), we also move the stop-loss on the remaining half to the entry price (break-even). This ensures the worst-case on the remainder is no loss (excluding slippage). - **Trail the Remainder using ATR:** After securing half, the other 50% of the trade is left to run with an active **trailing stop**. The trailing stop will be based on **1×ATR(something)** on M5 or M15. Specifically, at each new bar (or each certain interval), we set the SL at an ATR distance (e.g., 1x recent ATR) behind current price for longs (or above current price for shorts). This effectively follows the trend until it reverses by an ATR magnitude, capturing extended moves. - **Final TP at 2R:** If the trailing stop doesn't get hit earlier, we will cap the trade at +2R. That is, we set a take-profit for the remaining half at 2.0R profit. This is the "final target." If the price surges straight to 2R, we close the remainder there. If it meanders, the ATR trailing might stop it out somewhere between breakeven and 2R.

Justification: Option B is a more complex approach aiming to squeeze more profit from big trending moves while minimizing risk after an initial gain. By taking partial profit at 1R, we guarantee the trade won't be a loss once it's halfway to target (since half is banked and the rest is risk-free). The trailing stop on the remainder is to ride any strong trend beyond 1R – for instance, sometimes gold might run to 3R or more. This method could capitalize on that, whereas Option A would have exited at 1.5R. So Option B's advantage is **higher potential R per trade on big moves** and **no risk after hitting 1R**.

The downsides: it complicates trade management significantly (multiple actions: partial close, SL move, continuous trailing adjustments). It also can reduce total profit on typical wins – often gold might hit 1.2R then fall back to breakeven on remainder, so you end up with only +0.5R instead of +1.5R you'd have gotten by Option A. It relies on some trades trending far to compensate. It can also skew the distribution of outcomes, making many small wins and a few big wins, as opposed to consistent moderate wins.

We propose Option B as an **upgrade path** for later version (v2.1) once the base strategy is stable. It's an optional enhancement for users who want to maximize returns during strong trends. We will clearly separate it – v2.0 will ship with Option A logic active (no partials, no trailing). In v2.1, we can introduce an input toggle for "Advanced Trade Management" that switches to Option B rules.

Recommended Choice: Option A is recommended for v2.0 because of its simplicity and robustness: - It's easier to code and less prone to execution mishaps (partial closure in MT5 requires careful volume handling, trailing needs frequent position updates – more moving parts). - It aligns with our anti-overfit ethos by not adding many parameters (Option B adds at least: what ATR period? what ATR multiplier? what final TP? what partial ratio?). - It provides a consistent outcome structure that is straightforward to optimize for expectancy.

We believe Option A will yield a more stable performance across different market regimes. Once we gather data with Option A, we can evaluate if many trades would have greatly benefited from trailing. If, for example, we see many trades hit 1R then would have gone to 3R if not exited, we might then implement Option B carefully. But starting simple is prudent.

In summary, **v2.0 will use Option A: full TP at ~1.5R and full SL at 1R, with no partial or trailing.** This means if our strategy risk (SL) is 20 pips, we take profit at 30 pips. Every trade either loses 1 unit or gains 1.5 units (except rare cases like time-out or intervention). This clear rule will be coded and documented. Option B's multi-step exit will be prepared but likely commented out or turned off by default, to be enabled in future after further testing and justification.

(The user can of course experiment with Option B settings in the future; our EA code will be structured to allow adding such logic easily.)

5.3 Edge-Case Handling (MANDATORY)

Real markets throw curveballs. We need explicit rules for unusual but critical scenarios to protect the account. These edge-case rules are **mandatory** in v2 design:

- **Flash Crash Rule:** *"If an M1 candle > 500 pips → halt trading for 30 min."* As described, this protects against extreme events (e.g., a sudden massive spike due to a rogue trade or news). Implementation: On each new 1-minute bar, calculate its range. If `High - Low > 50.0` dollars (which is 5000 points if using point = $0.01, but the spec said 500 pips likely meaning $5.00 since often 1 pip = $0.10 for gold, but to avoid confusion we interpret it as 5000 points or $50? That seems huge. Possibly they meant 500 points = $5? We'll clarify: given context maybe 500 pips = $5.00. We'll use $5 move in 1 min as trigger). So if a 1-min bar exceeds a threshold (we'll choose the correct interpretation, say 500 points = $5), the EA sets a flag not to trade for the next 30 minutes. Essentially, `CanTradeNow()` will return false until `TimeCurrent()` > (flashCrashTime + 30 min). This rule prevents entering during the immediate chaos or erroneous prices. Such moves are exceedingly rare on gold outside of news; if it happens, best is to stay out until market calms.

- **News Filter:** *"Stop 5 min before major red events; Resume 15 min after."* We will maintain a calendar of major news (NFP, CPI, FOMC rate decisions, etc. – typically scheduled events known in advance). This can be input via an array of times or by reading an online calendar if advanced, but likely simplest is to provide the user input fields for a few known upcoming news times or let them manually avoid trading by disabling the EA around those times. However, since it's mandatory, we will build in basic logic:

- We'll have an input list (or maybe just a rule using keywords and the MT5 calendar functions, if available) for events considered high-impact. For each such event time, the EA will **stop trading 5 minutes before** the event. That means if current time is within T_event - 5min and T_event, `PreTradeGate` will block new trades.
- Any open trades could either be closed or managed specially. We choose to **flatten positions right before the event to avoid unpredictable slippage**, or at least tighten stop. But to keep simple, perhaps we just avoid new trades and let open trades run with their stops (the user can decide whether to close manually). Given 5 minutes is short, it might be prudent to close open positions 1-2 minutes before the event if they are in profit or small loss. But the spec doesn't explicitly say to close trades for news, just to stop and resume after. We might not force-close, but we will definitely not enter new ones.
- **Resume trading 15 minutes after** the event: big news often causes volatility for a few minutes. We'll implement a timer such that no new trades until T_event + 15min. This allows spreads and

volatility to normalize (as research also recommended ~5 minutes after open or news to let spreads settle [35]).

- This will apply to "red folder" events – we can define a static list (like NFP = first Fri each month 13:30 GMT, CPI US monthly, Fed interest rate decisions 8 times a year, etc.). The user might input them. For flexibility, we can have input e.g. `NewsFilter=true` and maybe an input string of times or simply advise user to not run EA at those times. A robust implementation might require using a news API which is outside scope. So we will implement a simplified approach: e.g., if the EA clock hits a known NFP time (we might pre-code known schedule if possible) – but that's not robust for all events.

- At minimum, we ensure the user can **manually pause trading** by a "News Filter" switch or by setting Trading_Day for that day false or such, if they know an event is coming. We'll document that. For automation, we might use the built-in Economic Calendar in MT5 (some APIs exist).

- **Weekend Handling:** *"Close all trades Friday 21:00 GMT."* This is straightforward. The EA will check the time on Friday. If it's >= 21:00 GMT (or maybe a few minutes before to be safe with execution), and we have any open position, it will close it. Additionally, we will ensure no new trades are opened on Friday after a certain cutoff (likely after Friday 20:55 GMT, for instance). This avoids liquidity drop and widening spreads in last hour of the week.

- The implementation might tie into `DayAllowed` or a separate check in OnTick: e.g., if `dt.day_of_week == 5 (Friday)` and `dt.hour*60+dt.min >= 21*60`, then for each open position with our magic, send a close order. We have to ensure to do it once, not repeatedly spam closes – so once closed, we're done.

- We also might set Trading_Day_Fri to false after 21:00 so that PreTradeGate blocks any entry (though at 21:00 we wouldn't anyway if we consider session ended at 18:00; but in case we allowed longer on Friday because New York close is actually 21:00 GMT in summer, we might have considered trading up to ~20:00. Either way, by 21:00 we definitely stop.)

- **Spread Spike Pause:** (Although not explicitly listed in point 5.3, it is an edge scenario we are covering in broker assumptions). If spread triples in <60s → pause 10 min. This is like a mini news or liquidity event filter. Implementation detail was discussed in section 3. It dovetails with news and flash crash but specifically triggered by spreads.

- The EA will track average spread over last 1 minute. If current tick's spread > 3 × (avg of last minute) and spread > some absolute threshold (like >50 already triggers anyway), then set a `spreadSpikeUntil = now + 10min`. `CanTradeNow` will return false until that time passes. This is to handle, say, sudden broker liquidity issues or unscheduled events.

- **Max Trade Duration:** *"3 hours (force exit)."* We will enforce that any open position will be closed after 3 hours if not closed already. Implementation: record the entry time of the trade (could store in `Signal.reason` or track via Position's time). We can check on each tick for any open position of ours: if `TimeCurrent() - position_open_time >= 3h`, then close the position at market. This is a safety to avoid turning a scalp into a swing or getting stuck. In practice, we expect few trades to last that long (our design has TP/SL likely hit by then). But if, for example, price got close to TP and then stagnated for hours, we'll just take whatever profit/loss at 3h mark and free capital.

- We should decide if 3h is counted strictly in market hours or can span overnight. But since we don't trade overnight, if a trade somehow was still open at session end, we'd close it Friday 21:00 anyway. On other days, if a trade triggered late in the session (say 16:00 GMT) it could still be open after 3 hours which would be beyond our session (19:00 GMT, after our normal trading window). That scenario is possible on Monday-Thursday if we allowed trades as late as 17:00 and they didn't hit exit by 20:00. Perhaps we should avoid that by not taking trades too late or by this rule which would close it by 20:00.
- Implementation corner: We must ensure to place this check perhaps in the `TM_ManageOpenPositions()` or a new similar manager function, so it closes without needing a new tick (though we will get ticks anyway as price moves a bit). It's okay to check on ticks. We will log that it was closed by max duration rule.

In summary, edge-case handling ensures: - The EA steps aside during **extreme volatility** (flash moves, news) to avoid unpredictable losses. - It doesn't hold trades during **poor liquidity times** (weekends, end of sessions). - It doesn't let trades linger beyond intended horizon (max 3h).

All these rules will be explicitly coded and are non-negotiable for safety. They complement our normal strategy logic to cover those rare scenarios that can otherwise ruin an account if not handled.

5.4 Risk Profile (Enhanced & Mandatory)

The risk management for v2 is comprehensive. In addition to per-trade risk (already covered as 0.25–1%), we impose several **risk limits** at the portfolio level:

- **Daily Loss Limit: 3% of equity.** In any single trading day, if the EA draws down 3% or more from that day's starting equity (or its peak equity of that day), it will stop trading for the rest of the day. This is akin to a daily stop-loss for the system. Implementation: We already have a daily DD tracker in v1 (Max_Daily_Drawdown_Percent = 5 by default). We will set it to 3 by default now. `TM_DD_Update()` tracks peak equity each day and `TM_DD_GateOK()` prevents new trades if current DD% >= Max_Daily_Drawdown [16]. We'll adjust that threshold to 3. This ensures if, say, we start the day with $10,000 and by mid-day we're down $300 (3%), we cut losses for that day. It prevents trying to "earn it back" under emotional or suboptimal conditions. It also aligns with prop firm style rules (they often have ~5% daily loss limit; we are even tighter at 3% by default to be safe).

- Note: If a great setup appears later that day, we'll miss it, but discipline says stop for the day when off track. We'd rather preserve capital and mindset for the next day.

- **Weekly Loss Limit: 7%.** Similarly, over a calendar week (Mon–Fri), if the equity drawdown reaches 7% from the week's starting equity peak, the EA will stop trading for the remainder of the week. Implementation: We need to track weekly performance. We can have variables for week start equity (set on Monday open or Sunday night) and track the peak equity during the week. If equity falls 7% from that peak, set a flag to halt trading until next week. This is not in v1, so we'll add it.

- Example: Starting week equity $10,000, peak reached $10,500, then a slump to below $9,765 (which is 7% down from 10,500), we stop. Even if some days remain, we take a break to avoid a deeper slide. Weekly limit being higher than daily (7 vs 3) allows a sequence of losing days but stops it short of serious damage. It's a sanity check and also somewhat replicates how professional risk managers limit weekly losses.

- **Max Consecutive Losses: 10 trades → pause 2 hours.** If the EA hits 10 losses in a row, it will stand down for a while (here 2 hours) to break the cycle. This rule recognizes that a streak of 10 consecutive losses might indicate either something is wrong with the strategy under current conditions or that market is behaving oppositely to our assumptions (perhaps a regime change). Pausing 2 hours (which likely means the rest of a session or until conditions change) is a cautious measure. Implementation: We will maintain a counter of consecutive losing trades. Each time a trade closes:

- If it's a loss, increment the counter.
- If it's a win (or even break-even), reset the counter to 0.
- If the counter hits 10, we set a `noTradeUntil = TimeCurrent() + 2 hours`. Also, we reset the counter (or maybe better, leave it until a win occurs? But after pause, presumably we start fresh).
- Then, `PreTradeGate` will check if `TimeCurrent() < noTradeUntil` and block trades if so.
- We will log that "Max consecutive losses reached – pausing trading for 2 hours".
- This rule prevents rapid-fire losses. For example, if something in the market fundamentally changed (e.g., suddenly every signal fails due to a regime shift or a technical issue), we don't keep slamming into losses. 10 in a row at 0.5% each is ~5% drawdown, so it also is a backstop before hitting those daily/weekly limits if they somehow didn't trigger first (10 losses might also trigger ~5% daily, but if spread across days, the consecutive rule catches it).

- The 2 hour pause is somewhat arbitrary but meant to skip the current session or wait for new info. We can adjust it as needed, but 2 hours is enough that conditions may reset (e.g., if it happened in London morning, maybe wait until NY open).

- **Margin Level Minimum: 200%.** The EA will check account margin level (equity-to-margin ratio) and stop trading new positions if the margin level is below 200%. A margin level of 200% means equity is twice the used margin – falling below that might indicate over-leverage or floating losses. We should never really get near 200% margin level because we risk small per trade. But if the user runs multiple EAs or if some external positions exist, margin could drop. By requiring >200% margin to open a trade, we ensure there's a healthy buffer. If margin level falls under 200% (maybe due to other positions' DD), the EA will not add new trades, and possibly could even close an open trade if we wanted to be super safe (but we'll at least not add risk).

- Implementation: Query `AccountInfoDouble(ACCOUNT_MARGIN_LEVEL)` each tick or on PreTradeGate. If not `EMPTY_VALUE` and < 200, block entries and log "margin level low".

- 200% is a common risk threshold – many systems and prop firms require it. It ensures at worst you'd have to lose 50% equity for a margin call (which we'll never get near in normal operation, but this is a safeguard).

- **Single Position at a Time:** Already mentioned – the EA will only open one trade at a time (per symbol). So maximum exposure is one position. This implicitly keeps combined risk low. If one trade is open with 0.5% risk, we won't have another adding additional risk. This is part of risk profile because multiple simultaneous trades could compound risk. Implementation: Before sending a new order, check `PositionsTotal()` for any position with our magic/symbol; or maintain a bool flag "inPosition". If found, skip new signal. (We will ensure to clear the flag when position closes; we might use trade events or just check positions length.)

- **Priority Instrument (XAU vs XAG):** The user specifically said *“If XAUUSD and XAGUSD signal → take only XAUUSD.”* This implies if two correlated signals come (like maybe a gold and silver strategy running together), prefer gold. In our single EA on gold, this situation doesn’t directly arise. But we can incorporate the spirit: if in future we expanded to multiple instruments or if the user attaches the EA to both XAU and XAG charts, we could implement a mechanism: e.g., a shared file or global variable such that if gold trade triggers, it prevents silver trade at the same moment. For now, we note this rule and ensure our design doesn’t attempt to trade multiple instruments concurrently. Possibly, we could add a check: if a silver trade is open (by detection of position on XAGUSD) and a gold signal comes, we may still take gold and maybe not allow silver if gold open – but that gets complex across charts.

- The safest interpretation: The user will likely run only one of these EAs at a time, or if both, they want the EA itself to choose. However, building inter-symbol communication is advanced. Maybe in v2.0 we won’t implement cross-instrument logic beyond stating: “We focus on XAUUSD – if user also runs on XAGUSD, they should set it to trade only if no gold trade open (which could be done via some shared flag).”
- Possibly we introduce an input like `AllowTradingIfOtherSymbolPosition = false` and use a static shared memory (like GlobalVariable or file). But this might be out-of-scope for now. We’ll mention it as a guideline rather than coding it fully in v2.0.

All the above risk limits will be **hard-coded or controlled via inputs** and enforced by the EA logic. They ensure that even in worst-case scenarios, the account is protected: - A disastrous day can cost at most ~3%, - A disastrous week at most ~7%, - A freak sequence of losses stops out after ~5% and forces a pause, - The EA never over-leverages or stacks positions.

These measures complement the per-trade stop losses. They act as a **second layer of defense**. While ideally the strategy doesn’t hit these limits often (if at all), having them is crucial. They make the system robust to unexpected conditions or performance degradation. The user can also adjust these thresholds (some may prefer a 5% daily, or a 10% weekly – we’ll allow some tuning, but defaults are 3/7 to be conservative).

In summary, the risk profile of v2 is stringent: **small fixed fractional trade risk, plus overall loss caps, plus trade frequency limiters in losing streaks, plus margin safeguards, plus single-trade exposure.** This ensures longevity of the trading account and confidence in the system’s behavior under stress. Even if performance is not as hoped, these mechanisms prevent catastrophic losses, giving time to recalibrate or stop if needed. This is a non-negotiable aspect of the design; it’s baked into how the EA operates daily and weekly.

With points 1–5 covered, we have the blueprint for *what* the strategy should do. Next, we translate that into the actual strategy rules (point 7), but first, we outline how we will choose and optimize the strategy’s parameters (point 6).

6. Parameter Optimization Plan

We will have a number of **parameters** in the v2 strategy – things like indicator periods, thresholds, risk settings, etc. For each parameter introduced, we will specify: - A **default value** that is a reasonable starting point (likely from known practices or initial tests), - An **optimization range** (min, max, step) for testing different values in backtest, - Whether it's **static or dynamic**, i.e., fixed value or changes based on market regime, - And a note on **stability** – the parameter should work across multiple market regimes (trending, ranging, high volatility) without needing change; if not, we consider making it dynamic.

Below is a table of key parameters we anticipate and their plans:

1. **Trend EMA Period** – *Definition:* period for the EMA used as trend filter (if we use one).
2. **Default:** 50 (on M5). A 50-period M5 EMA roughly corresponds to a 50*5min = 250min ~ 4-hour trend, which is a good intraday trend gauge. We could also use 100 or 200; v1 used 200. We'll test between shorter and longer.
3. **Range:** 50 to 200, step 25. We'll try 50, 75, 100, 125, 150, 175, 200 in optimization to see which yields best results. Possibly the optimum lies in 50–100 range because 200 was likely too slow.
4. **Static vs Dynamic:** Static. We will not change the EMA period dynamically – it's a baseline measure of trend. We might later adjust how strict we use it (e.g., maybe dynamic threshold like using it only when ADX high?), but the period itself stays fixed.
5. **Stability:** It should be effective across regimes. If the market regime changes (trending vs ranging), the EMA filter might sometimes filter out too many trades in ranging markets. But that's intended. If we see that in heavy ranging periods the EA sits out entirely because of EMA, we might allow a dynamic rule (like if ADX very low, perhaps ignore EMA and take both directions because no trend). For now, we set it static (e.g., 50).

6. We ensure whichever default we pick works reasonably on both trending months (won't whipsaw too much) and ranging months (still allows some trades, perhaps via other logic).

7. **ADX Period & Threshold** – *Definition:* period for ADX and threshold to consider "trending." We might use ADX to decide dynamic behaviors (like altering ATR multiplier or whether to apply trend filter strictly).

8. **Default Period:** 14 (standard). **Default Threshold:** 20. ADX 20 is commonly used as the line between low and moderate trend. v1 used threshold 25 for regime split [39]. We may lower to 20 to be a bit more sensitive, since 25 was maybe too high (cutting out moderate trends).
9. **Range:** ADX period 10–20 (step 2); threshold 15–30 (step 5). We'll test combinations, possibly focusing on threshold because period 14 is usually fine.
10. **Static vs Dynamic:** The ADX itself is dynamic measure of market. We will use it dynamically – for example, if ADX < threshold, we might loosen some conditions (like perhaps allow counter-trend or use smaller ATR stops; if ADX > threshold, tighten up or only trend-follow). The threshold itself will be static number (like 20). So ADX usage introduces dynamic behavior by design: it detects regime changes.

11. **Stability:** ADX's meaning (trend strength) is fairly consistent across markets. We'll ensure the chosen threshold works in both quiet and volatile times. If we see ADX staying high in certain volatile ranging scenarios (like whipsaw volatility can keep ADX artificially high), we might be cautious in over-relying on it. We'll likely use it in a minor way: e.g., if ADX is very low, maybe skip breakout trades; if very high, maybe skip reversal trades. The default 20/25 threshold has been used for decades in TA, which is a good sign of stability. We won't overtune this to a narrow value.

12. **RSI Period & Thresholds** – *Definition:* RSI period for entry signal and the oversold/overbought levels.

13. **Default Period:** 14 (classic). We could consider shorter like 8 or 10 for faster signals on M5, but 14 is a stable default across many strategies (v1 used 14).
14. **Thresholds:** Default buy threshold 30 (below 30 = oversold), sell threshold 70 (above 70 = overbought). V1 had 35/70 [40] (35 is slightly less strict for oversold than 30). We might use symmetric 30/70 or even 20/80 for stronger extremes. But 30/70 is standard.
15. **Range:** Period 8–21 (we'll test a few values: 8, 14, 21). Buy threshold 25–40, Sell threshold 60–80. We will maintain a gap (Sell thr > Buy thr obviously). We'll likely optimize one side then mirror it, or optimize both if needed.
16. **Static vs Dynamic:** Static. We won't change RSI period dynamically – it's a fixed indicator. The threshold might effectively be dynamic if we tie it to other conditions (e.g., maybe require a deeper oversold if no other confirm, but allow milder oversold if with pattern – but that's complicating. We prefer static threshold). So likely keep it fixed (like 30/70).

17. **Stability:** RSI extremes have historically worked similarly in various regimes – extreme oversold often precedes bounces even in volatile times. However, in strong trends, RSI can stay oversold/overbought for long. That's why we combine RSI with a trend filter: in uptrends, we trust oversold signals, in downtrends oversold can be ignored because it can stay oversold (i.e., don't counter-trend just on RSI). So stability comes from using RSI in the right context. The default 30/70 is well-known and should be broadly applicable. We'll verify that tweaking to 35/65 or 20/80 doesn't drastically improve things – likely not by much.

18. **ATR Period & ATR-based Stop Multiplier:**

19. **Default ATR Period:** 14 (to match typical usage and v1). Possibly 20 if we want a longer measure, but 14 is fine.
20. **Stop Multiplier:** Default roughly 1.5 for trend trades (as in v1) and maybe 1.2 for mean reversion. But since we are leaning to one unified approach, we'll pick a single number or a small set. Perhaps **1.3 ATR** as a baseline SL distance. Actually, given we plan to target 1.5R, our TP ~1.5*SL – if SL = 1.3 ATR, TP = 1.95 ATR. That seems reasonable. We'll fine-tune though.
21. **Range:** ATR period 10–20. Multiplier 1.0 to 2.0 (in steps of 0.1). We expect the optimum to cluster around 1.2–1.6 based on v1 and general practice. Too low (1.0 ATR) might stop out too often on noise, too high (2 ATR) might be too wide reducing reward.
22. **Static vs Dynamic:** We can make this **dynamic** relative to ADX or regime: e.g., *if trending (ADX high) use 1.5, if ranging (ADX low) use 1.0*. Or *if volatility very high (ATR itself is large in absolute terms), maybe use a smaller multiplier to reduce absolute stop – although ATR inherently scales it.* Possibly more useful: *If we identify a setup as mean reversion vs breakout, we could assign different multipliers.*
    - For initial simplicity, we might set one multiplier ~1.3 for all trades, or perhaps two: one for long trades one for short? Probably not necessary.
    - However, given v1 did differentiate ATR_Mult_SL_Trend vs ATR_Mult_SL_Range, we can carry that idea: optimize two values – maybe results say e.g. 1.4 for trends and 1.0 for counter-trend works best. We'll include that possibility (so dynamic by regime).

23. **Stability:** We need the stops to be effective in both low and high volatility. Using ATR helps since ATR rises in high vol (so stop widens automatically). The multiplier might not need change often. Maybe during extremely choppy periods a larger multiplier protects from whipsaw, but that also reduces R. We will likely keep it fixed or at most two fixed values for two regimes, which is stable enough. The exact default chosen (say 1.3) will be based on tests across different market conditions (we'll test on a trending period and a ranging period to ensure it's neither too tight nor too loose in either).

24. **Take-Profit Multiplier (R) if using ATR or fixed pips:**

25. Actually, since we defined TP in terms of R (1.5R by default), we don't need an absolute value. The SL and TP are linked by R ratio. So main parameter is the target RR ratio.
26. **Default RR_Target:** 1.5. Range: test 1.2, 1.5, 2.0 maybe. 1.5 is recommended, but we might verify if 1.4 or 1.6 performs better. (We suspect 1.5 is fine).
27. **Static vs Dynamic:** We keep RR static. Changing target on the fly would be overfitting (like "if last trades lost, aim for 2R to recover" – no, we don't do that). However, Option B would effectively introduce dynamic partial exits – but as per plan, we not doing that in v2.0.

28. **Stability:** The chosen RR should yield good risk-reward in various conditions. If the market is more mean reverting, smaller RR might work better (take profit sooner). If trending, larger RR yields more profit. 1.5 is a middle ground. We will confirm that strategy doesn't break if we adjust it by ±0.3. If results are similar for 1.3–1.7, it means it's not overly sensitive – good stability. If it's very peaked, we must be careful, but usually it's not that sensitive.

29. **Session Time Inputs:**

30. **Default:** Start 07:00 GMT, End 18:00 GMT (these are from v1 original).
31. We'll provide them as inputs (`Session_Start_Hour`, `Session_End_Hour`) in case user needs to tweak (some might prefer 06–17 or 08–20 depending on DST or broker timezone). But default covers London+NY overlap well.
32. **Range:** 6–9 for start, 17–21 for end (we won't "optimize" this in backtest since it's discrete and environment-based, but it's adjustable).
33. **Static vs Dynamic:** Static daily schedule. We won't change hours dynamically. (Though perhaps if high impact news at 14:00, one could dynamically skip that half hour, but that's covered by news filter, not session hours).

34. **Stability:** The 07–18 GMT window is historically when gold has volume; this has been true across years. We include a bit of pre-London (7 GMT is 1 hour before official London open at 8 GMT, capturing some early moves) and stop before or around when US afternoon quiets down. This should be stable. If needed, user can shift by 1–2 hours without major impact.

35. **News Filter toggles:**

36. We'll have `UseNewsFilter` (bool). Default true (assuming we do have a mechanism).
37. Possibly inputs for specific news times (could be a list of HH:MM or a string; but implementation might not be straightforward to optimize – likely not optimized, just user setting).
38. Range: On/Off for filter; (if off, EA trades through news – not recommended but user's choice).
39. Static vs Dynamic: It's static setting; the events themselves occur dynamically but the rule is static.

40. Stability: Always skipping major news is generally a stable rule – rarely would trading *through* news be advisable. So we keep this on by default.

41. **Risk Settings:**

42. `Risk_Percent` default 0.5%. Range 0.25–1.0%. (User can choose; we won't optimize this in the strategy, as it's more of a user preference/trade-off. We might test performance at 0.5% to ensure DD in target.)
43. `Max_Daily_DD` default 3%. Range 2–5%. (Not optimizing, just user parameter.)
44. `Max_Weekly_DD` default 7%. Range 5–10%.
45. `Max_Consecutive_Losses` default 10 (int). Range 5–15.
46. We won't treat these as optimization variables; they are set by risk tolerance. They must be static during execution (though a user could adjust if they found 3% daily was too tight and strategy rarely hits it – maybe up to 4%. But not something an algorithm picks; it's risk appetite).

47. Stability: They are part of risk framework; as long as they're not hit often, they won't interfere. They are deliberately conservative. If market regime changes cause more frequent hitting, user can adjust upward slightly, but we aim not to.

48. **Miscellaneous:**

49. `MagicNumber` offset (just an integer, not optimize).
50. `Max_Spread_Points` default 50. Range maybe 30–70 if user wants. No need to optimize; user sets if broker conditions differ.
51. `Max_Slippage_Points` default 50 (like v1). Range similar, user adjustable.
52. `Order_Expiration_Min` for pending orders: default 30 min (from v1). If we use stop orders for breakouts, they cancel after 30 min if not filled. We might keep or even shorten if we rarely use. It's minor.
53. `Only_New_Bar` default true (trade on new bar). We likely keep this logic to evaluate signals on bar close, which helps prevent whipsaw entries mid-candle. We might test false (allow intra-bar) but likely true is simpler and avoids noise. If we do high frequency, we might consider false, but given our moderate frequency, per-bar is fine. Keep as input though.

Parameter Stability across Market Regimes: We will test the chosen defaults on: - A **trending period** (e.g., a strong bull run in gold) to ensure the trend filter and ATR stops work (should catch pullbacks and not exit too early). - A **ranging period** (e.g., low volatility, mean-reverting phase) to ensure it still finds trades (some mean reversion trades maybe) and doesn't over-risk (the ATR filter might skip if too low vol, but hopefully not all trades). - A **high-volatility event** (like 2020 March COVID volatility or a war headline) to see if stops held and risk limits triggered appropriately. While parameters like ATR multiplier might ideally be dynamic, we expect our combination (ATR scaled stops + small risk + halts) to handle it. If we see that ATR multipliers needed different values in calm vs volatile periods, that indicates lack of stability – to fix that, we could incorporate ADX/ATR dynamic adjustment. For example: if ATR (volatility) is extremely high (say ATR(14) > some threshold), maybe reduce position size or skip some trades. Or contrarily, if volatility is high, keep trading but risk per trade effectively goes down because each trade's pip risk is bigger relative to equity (if using % risk, it auto-adjusts lot size smaller).

The plan is to choose parameter defaults that yield profitability **in at least 3 distinct market regimes**: 1. **Trending market:** (e.g. sustained uptrend or downtrend) – strategy should produce a series of wins (taking pullbacks or breakouts in trend). 2. **Choppy sideways market:** – strategy might trade less or have more mixed results, but risk controls should keep losses small. If too many losses, maybe a filter (like ADX low -> maybe don't trade breakouts) will mitigate it. 3. **High volatility swings:** (like big daily ranges without clear trend) – strategy should either cautiously trade or skip if conditions are beyond its scope. Our flash crash/ news rules skip the wildest times. If moderately high vol (ATR double normal), our ATR-sized stops automatically widen, so trades can survive larger swings. We ensure the profit target 1.5R doesn't become unreachable (if ATR stops are huge, 1.5R is huge too, maybe won't hit in volatile whipsaw – but presumably if ATR is huge, moves can hit it).

We will confirm in backtests that the EA is **profitable across 3+ different market regimes**: - E.g., year 2019 (range-bound gold), 2020 (wild swings, up then down), 2021 (strong trend up), etc. Or just using statistical regime segmentation by ADX. We might run a backtest segmented by ADX regime to ensure it isn't only profitable in one regime. - If we find a parameter is only good in one scenario, we either adjust it or introduce dynamic behavior triggered by a regime indicator (like ADX). For example, if a short ATR period works well in trends but gives false signals in ranges, we might keep it but rely on ADX to filter when to trust those signals.

To sum up: - We identified all crucial parameters, - Provided initial default values (from common practice or v1 experiences), - Defined reasonable ranges to explore in optimization (to fine-tune for best results without going crazy), - Marked which will be dynamic (like ATR multiplier by regime possibly) and which static, - And emphasized each must be vetted for multi-regime robustness.

This plan ensures our final settings are **not cherry-picked for one period**, but broadly effective. It also means the EA has a moderate number of tunable inputs (we're looking at maybe 10-15 user inputs, many of which are risk/session toggles rather than curve-fit numbers). This keeps it user-friendly.

Next, we proceed to the actual strategy redesign (point 7), using these parameters and rules to formulate the new trading logic.

7. Complete Redesign of the Strategy (v2)

Based on the research, constraints, and targets above, here is the **full conceptual strategy for v2** – detailing all components: trend filter, volatility filter, entry logic, exit logic, risk management, sessions, spread/slippage rules, news protection, timeouts, etc. This is the blueprint that our implementation will follow to naturally achieve ~100–300 trades/year, with robust performance.

Overall Strategy Concept:

In essence, v2 is a hybrid scalping strategy that combines trend-following and mean-reversion elements depending on market conditions: - When a clear intraday trend is present, it scalps **pullbacks** in the direction of the trend. - When the market is range-bound or choppy, it scalps **reversals** at extremes of the range (mean-reversion). - It avoids trading counter to very strong trends and avoids low-probability breakouts during dead markets. - Entries are confirmed by a confluence of a few simple signals (trend bias + short-term oscillator extreme + price pattern). - Exits are predetermined (SL and TP at 1R/1.5R respectively) or time-based, with no discretionary elements.

The strategy can be thought of in two modes that it seamlessly switches between: 1. **Trend Scalping Mode:** (Activated when market shows momentum, e.g., ADX is high or price above EMA) - Only trades in the direction of the prevailing trend (e.g., only buys if uptrend). - Waits for a pullback: an oversold condition in that uptrend (e.g., RSI dips below threshold) and a minor price pattern indicating resumption of the trend. - Enters with a tight stop below the pullback low, aiming to catch a quick continuation move. 2. **Range**

Scalping Mode: (Activated when market is flat/slow, ADX low) - No strong trend, so it can trade both directions, but only at extremes. - Identifies overbought conditions near range highs or oversold near range lows, and enters expecting a reversion toward the mean. - Confirms with a candle pattern showing rejection of the extreme (e.g., a wick or engulfing). - Takes a small bite out of the ensuing move, with tight stop just beyond the extreme point.

Both modes share many components; the difference is mainly whether we require trend alignment or not. The EA decides mode dynamically using a **Trend Filter (EMA & ADX)**: - If price is significantly above the EMA and ADX is above threshold (trending), use Trend Mode (only look for longs on dips). - If ADX is low (non-trending or very mild trend), allow Range Mode (consider both longs and shorts at extremes). - If ADX is high but we have a contrarian signal, likely skip it because it's dangerous to fade a strong trend.

Now, breaking down the components:

Trend Filter Rules: - Use a **50-period EMA on M5** as the baseline trend indicator (parameterized, default 50). Also consider the **200-period EMA** as a higher-level trend if needed – but likely 50 is enough for intraday. - And/or use **H1 price action**: we could say if the current price is above the H1 20 EMA (for example), that's a bullish bias. But to keep it simple, we stick to M5 EMA which roughly reflects that. - **Rule:** If price (M5 close) is above the EMA50 and the EMA50 itself is sloping upward (we can check EMA50 > EMA50 of X bars ago), then bias = Bullish. If below and slope down, bias = Bearish. If price whips around EMA and slope is flat, that's no clear trend (likely ADX will be low too). - Additionally, check **ADX(14)** on M5: If ADX >= 20 (moderate trend strength), we trust the trend bias. If ADX < 20, we treat it as ranging even if price is a bit above/below EMA (because that could be noise). - So effectively: - **Trending condition:** ADX >= 20 AND (price above EMA50 -> bullish bias; price below EMA50 -> bearish bias). In this state, we will **only trade in the bias direction**. Counter-signals will be ignored (or possibly taken only if extremely good – but safer to ignore). - **Ranging/Weak trend condition:** ADX < 20 (regardless of EMA50). In this state, trend filter is relaxed – we allow trades either direction, but we ideally want to see price at an extreme relative to some band (like Bollinger or simply far from EMA). - If ADX is marginal (e.g., ~20) and EMA slope is unclear, we may consider it ranging to be safe.

This trend filter ensures we don't short a strong uptrend or buy a falling knife. It increases trade frequency in ranges by allowing both sides, and cuts frequency in trends by disallowing counter-trend trades (which are likely losers in strong trends).

Volatility Filter Rules: - Use **ATR(14) on M5** to ensure sufficient volatility but not too much. Specifically: - Require ATR(14) in points >= a minimum (Min_ATR_Points). For XAUUSD, from v1 config, Min_ATR_Points = 100 (~$1.00). We'll keep a similar threshold. If ATR(14) < 100 points (meaning last 14 bars had average range < $1), that's extremely quiet – likely not worth trading (spreads and noise would eat profit). So we will **skip new entries when ATR is too low**. - Conversely, if ATR is extremely high (like > some value), we still trade but our stops/tp scale with ATR anyway. However, if ATR is so high that 1.5R target becomes huge (maybe news environment), we rely on the news filter/flash crash rules to avoid it. So no explicit upper bound on ATR, aside from the fact that if ATR skyrockets, probably ADX or spread will trigger other protections. - Additionally, a **Volume filter** can be considered (like v1 had `VolumeConfirm`). But volume in spot FX is tricky; if we had futures volume, we could require a volume surge to confirm breakouts (v1 did for breakout trades [41] [42]). Given spot volume is tick count, we might not use it now except possibly for breakouts. We likely skip volume filter due to reliability concerns in spot. Instead, ATR already proxies volatility.

Entry Logic:

We will now describe how a **signal is generated** (functionally what `BuildSignal()` will do):

On each new M5 bar (during allowed sessions, and if PreTradeGate passes spreads/limits):

1. **Determine Regime:** Calculate ADX(14) and trend bias (EMA50). Set a flag `trend_mode` = true if ADX>=20 and clear EMA bias, else `trend_mode` = false (range mode).
2. Also identify `bias_dir`: if trend_mode true, bias_dir = Long or Short depending on EMA; if trend_mode false, bias_dir = None (meaning willing to trade both).
3. **Identify Potential Entry Direction(s):**
4. If `trend_mode` is true (say bias_dir = Long):
    - We will **only consider a Long trade** setup this bar. Short setups are ignored.
    - Entry trigger will require a **pullback and resume** in that direction.
5. If `trend_mode` is false (range mode):
    - We consider **both long and short** setups, but typically one will appear at a time (depending if price is at a high or low extreme).
    - We need to gauge if price is at an upper extreme (then look to short) or lower extreme (then look to long). We could use an indicator like Bollinger Bands (20,2). For example: if price closed outside the upper Bollinger band, that's an extreme overbought -> potential short. If below lower band -> potential long. Alternatively, use RSI: if RSI > 70 and perhaps price above EMA (but ADX low), that suggests overbought -> short. If RSI < 30 suggests oversold -> long.
    - We can also keep track of a recent range high/low. For simplicity, we can approximate extremes by RSI and perhaps distance from EMA.
6. **Oscillator Condition (RSI):**
7. Compute RSI(14) for the last closed bar (or current bar if we want intra-bar? But we are doing on new bar so last closed bar's RSI).
8. If looking for a Long entry: require **RSI(14) <= 30** (or threshold input, default ~30-35). This indicates oversold or at least on the lower end of range.
9. If looking for a Short entry: require **RSI(14) >= 70** (or threshold ~65-70).
10. These ensure we enter when price is somewhat stretched opposite the intended trade direction (pullback in uptrend or extreme in range).
11. If RSI condition isn't met, no trade signal this bar.
12. *Note:* In a strong trend, RSI might not drop to 30 before trend continues (e.g., in uptrend, pullbacks might only push RSI to 40 then price continues up). To not miss trades, we might have to use a slightly higher threshold in trending context (like allow RSI <= 50 in a confirmed uptrend?). But that could give too many signals. Perhaps compromise: in trend_mode, we use a moderate threshold (say 40) just to ensure some pullback, whereas in range_mode we use extreme (30). We can make RSI threshold dynamic: if bias_dir is set (trend_mode) use one value, if not (range) use another. For now, default 30/70 should catch strong pullbacks only; we can test if we need to loosen for trend trades.

13. **Price Pattern Confirmation:**

14. We need evidence that the pullback or extreme is ending and price is turning in our favor.
15. We will look for a **bullish reversal candle pattern for longs** or **bearish for shorts** on the just-closed bar or the bar before:
    - Examples: an **engulfing** candle (e.g., for long: the last bar is bullish and its body engulfs the body of prior bar which was bearish), or a **hammer/pin bar** with long tail down, or simply **two-bar pattern** like a big down bar then a big up bar.
    - To keep coding simple, we might implement one pattern clearly:
    - **Engulfing rule:** For a long: the latest closed bar's body is green and its open is below the previous bar's close and its close is above previous bar's open (engulfs previous body). And previous bar might be red or small. For a short: last bar is red body engulfing prior body.
    - Alternatively, **close vs previous close:** A simpler check: require that the latest bar closed **higher than the high of the previous bar** for a long signal (i.e., a bullish momentum bar after the pullback). That was how v1's breakout logic confirmed breakouts (close beyond a level) [8] . We can use a similar idea: for long, if last close > max(last 2 bars' highs), it suggests buyers regained control – that's a breakout of the immediate minor resistance.
    - For short, require last close < min(last 2 bars' lows).
    - We also want to ensure there was a pullback prior: e.g., at least one of the previous bars was bearish (for a long setup). RSI being low hints that, but also price pattern: perhaps check the prior bar or two were red or had lower closes.
    - So one concrete implementation:
    - For Long: If (RSI <= 30) AND (previous 1 or 2 bars had a lower close than their predecessor – indicating down movement) AND (the last bar closed up and above the high of those prior 2 bars).
    - For Short: If (RSI >= 70) AND (previous bars showed some up movement) AND (the last bar closed down and below the low of prior bars).
    - This effectively creates a mini "fracture" signal: price made a minor low and then broke a minor swing high.

16. If this pattern condition is not met, we do not enter, even if RSI was extreme. This filter will remove a lot of false signals where RSI is oversold but price keeps falling or just goes flat. It ensures momentum has flipped in our favor.

17. **Spread & Execution check (Pre-trade gate):** Even if a signal passes all of the above, we do one more check right before entry:

18. Ensure current spread <= Max_Spread (15–50 points as set) [37] .
19. Ensure no news upcoming in next 5 min (the News filter will have set a flag if so).
20. Ensure not in the paused state (due to daily loss, weekly loss, etc. – trade gates cover those).
21. If any fails, we abort the signal (don't trade this bar).

22. If all good, proceed to send order.

23. **Entry Price and Order Type:**

24. We will typically enter at **market price** once a new bar opens (because our signal is based on closed bar confirmation). So essentially at the open of the bar following the signal bar.

25. Reference price for SL/TP calculation will be this entry price (or we can use last close if similar).
26. In some cases, we might consider a limit entry if we expect a slight retrace (like enter on small pullback instead of immediate open). But to keep it simple and not miss trades, market entry on open is fine – slippage is limited by our max slippage setting (50 pts).
27. If our pattern was a break of previous high/low, often the entry might be a few pips above that breakout. But since we execute at open of next bar, which likely is at or slightly beyond that breakout level if momentum continues, it's okay. If next bar opens much higher (gap up), we might get slippage – but our slippage control will prevent a bad fill or skip if too much.

28. So implement as `ExecuteMarketOrder` (MT5 instant) with a comment reason code "SCALP_LONG" or "SCALP_SHORT".

29. **Stop Loss placement:**

30. For a Long trade: place SL just below the recent swing low of the pullback. Concretely, that could be:
    - The low of the **signal candle** or the **previous candle**, whichever is lower. For instance, if an engulfing pattern: the low of the pattern (which likely is the prior bar's low if that was the pullback bottom).
    - We can also use ATR: Compute ATR(14) in points and set SL = entry_price - (ATR * 1.3) for long. But price structure is often better for precise stops.
    - A compromise is to take max of (structure-based distance, ATR-based minimum). For safety, ensure SL is at least `MIN_SL_POINTS` away (like 10 points to avoid invalid stops).
    - Likely approach: **SL = min(low of last 2 bars) - buffer** for a long. Buffer could be a small fixed 5-10 points or 0.1*ATR, just to avoid stop exactly at swing low (to reduce stop-hunting risk).
    - This typically yields a stop maybe on the order of 20-40 pips in many cases (depending on pullback depth).
31. For a Short trade: place SL just above the recent swing high (high of last 1-2 bars) plus a small buffer.
32. Also cross-check this SL distance in pips vs ATR * multiplier:
    - If structure-based SL is larger than ATR*multiplier (1.5), that might indicate a very big pattern – maybe skip trade or adjust? If it's far, our position size calculation will reduce lot (so it's okay but maybe risk:reward might become less than 1.5 if our TP logic is still 1.5 * ATR).
    - If structure SL is much smaller than ATR*mult (like a very tight stop in volatile moment), that might be too tight and likely to get hit. We will enforce a minimum ATR fraction. Actually the code can be simpler: use ATR stop always for risk calc, but structure stop for order – not ideal, better to unify. Alternatively, we pick whichever is larger to avoid too tight: `sl_points = max( structure_points, ATR*1.0 )`. But structure normally should be smaller if it was a small pullback.
    - Let's do: **Final SL distance = max(structure_distance, 0.8*ATR)**, to ensure not ridiculously tight. This factor can be tuned. This ensures in volatile conditions where structure might be noisy, we give at least some breathing room.

33. Once SL price is set, calculate TP price relative to that (for Option A).

34. **Take Profit placement:**

35. Using Option A: **TP = entry_price + 1.5 * (entry_price - SL)** for long. For short, TP = entry - 1.5*(SL_distance).
36. This ensures R:R = 1:1.5.

37. We will also enforce a minimum TP distance (like MIN_TP_POINTS, maybe 10 points) just to avoid too close (shouldn't happen normally given stops).
38. If that TP is extremely far relative to recent volatility (like more than what usually happens in a session), we rely on either trailing stop or time exit eventually. But 1.5R shouldn't be too far – if the pattern was valid, often price can move that much in next 1-3 bars or so in a good scenario. If not, then maybe our trade fails or times out.

39. For partial/Option B (not in v2.0, but to note): no need now; currently entire volume targets this TP.

40. **Risk Management on Entry:**

41. Compute position **lot size** such that if SL is hit, we lose `Risk_Percent` of equity.
42. Use the `ComputeLot()` function similar to v1 [13] [43]:
    - Input: entry price, SL price.
    - It figures out distance in points and uses tick value to get how much per lot that distance costs, then calculates lot = money_risk / (distance * value_per_point).
    - We cap lot to min and max volumes as per symbol spec.
43. This ensures each trade is equal risk. If our SL is bigger, position size is smaller and vice versa.
44. If `ComputeLot` yields 0 (e.g., if required lot is below broker min), then the trade might be too small to take – we'll log and skip it (for a large account that's unlikely).
45. We also ensure margin is enough (though margin check is separate).

46. After compute, we round lot to 2 decimals or instrument's allowed precision.

47. **Final Entry Execution:**

    - If all above have signaled a trade, we send the order:
    - Type: Buy or Sell (market).
    - Volume: as computed.
    - SL: set as computed price.
    - TP: set as computed price.
    - Slippage: allow up to Max_Slippage_Points.
    - Comment with reason: e.g., "SCALP_L" or "SCALP_S" for long/short (to identify in logs).
    - If order is filled, we log the entry. If it fails (e.g., no liquidity, or slippage too high triggered rejection), we log the failure. If fail, we do not retry immediately (we skip this trade).
    - Now one position is open. EA won't open another until this closes (one at a time rule).

Exit Logic:

Once in a trade, we manage exits as per Option A rules primarily:

- **Stop-Loss Exit:** If price hits the SL level, the trade closes at a loss of ~1R (or slightly more if slippage – but we hope slippage minimal except maybe news times which we avoid). The trade manager or built-in MT5 SL will do this automatically – we set a hard SL with the order, so MT5/broker stops it out.
- **Take-Profit Exit:** If price hits the TP level, the trade closes with +1.5R profit. Also handled by broker since TP is set.

- **Time-based Exit:** We will implement a check to close the position if it's been open for >= 3 hours (180 minutes). This will be done in the OnTick manager:
- If `TimeCurrent() - position_open_time >= 10800 seconds`, then close at market.
- This might close trade at a smaller win or loss than planned. We consider this a graceful exit – maybe the trade stagnated. We log "Max duration reached – closing".
- This ensures no trade lingers overnight by accident (since we don't trade overnight; but if one did, 3h would close long before Asia close).
- **Session cutoff exit:** If it's Friday 21:00, we close as discussed. Also, possibly if we hit end of session (18:00 GMT) and we prefer not to hold after session – we didn't explicitly say we'd close at session end on other days. Typically, it's okay to hold a trade opened at 17:55 GMT until 19:00 if TP/SL not hit, but liquidity after 18:00 is less. We haven't mandated closing everyday at 18:00, just not opening new. We might allow it to run for up to 3h rule which would close by ~20:55 anyway if opened at 17:55. That covers it naturally. So no daily closure except Friday.
- **Manual or external exit:** If user manually closes or a stop-out due to margin occurs (unlikely with our margins), our code should handle it. We'll detect position closed and reset tracking variables accordingly (like consecutive loss count update).

Since we're not doing partials or trailing in v2.0, there's no dynamic exit management needed beyond what's above. However, we will incorporate certain *management tools*: - **Break-even move (if we had Option B)**: Not used in v2.0, but we keep the code there disabled possibly. (Trade manager in v1 had `Move_BE_After_Points` etc [44], but by default we can leave those off.) - **Trailing stop**: Also in trade_manager (Trail_Mode). We will set `Trail_Mode=0` by default (off) to align with Option A. So no trailing will actually happen. The code is there but inactive. - This keeps things simple: essentially a fire-and-forget trade with SL/TP.

Risk Management & Guards Recap (during trade): - **No adding to position**, no martingale or anything. Only one entry, one exit. - **No moving SL/TP** manually (Option A). We do not adjust them unless partial/trailing which we are not doing now. So initial SL/TP remain fixed (except time closing earlier). - **Consecutive loss counter:** After a trade closes (via any exit): - If it closed at SL (or at a loss overall), increment the loss_streak counter. - If it closed at TP (profit) or even closed positive at time exit, reset loss_streak to 0. - If loss_streak reached 10, set the 2h pause (noTradeUntil). - These are handled likely in the trade_manager loop or in OnTick after closing detection. We'll implement by checking closed trades list or using history (MT5 allows HistorySelect and HistoryDealGetProfit for last trade, or we track outcome in trade_manager partial). - This ensures the EA might take a break if many losses in a row (the condition to resume is after 2h or maybe after a winning trade, but we specified 2h anyway). - **Daily/Weekly loss gates:** Continuously updated by `TM_DD_Update`: - If daily DD >=3%, `TM_DD_GateOK()` returns false, so PreTradeGate will block new trades [16]. - If weekly DD >=7%, we will implement similarly (perhaps extend trade_manager to track weekly). That would also cause PreTradeGate to block trades for rest of week. - These gates do not automatically close existing trades (we could optionally, but currently it just stops new ones). We rely on stops to handle open ones. Arguably, if daily loss limit hit and a trade is still open in loss, maybe we should close it to prevent further loss. But that's a bit tricky (closing at peak loss might backfire if it reverses). Many prop rules would require closing though. For caution, we could decide: if hitting daily loss, do not open new and perhaps tighten stops on open trade or something. Simpler: if an open trade's SL is beyond the daily limit, that means if it hits SL we overshoot daily 3%. But our sizing likely set that exactly, so one SL = 0.5% of equity, so we'd need 6 losses to hit 3%. So if we are approaching daily 3%, probably no trade open or maybe a final one small. We won't force-close any open due to daily rule in this design, just stop adding new. - **Margin level guard:** On each OnTick, if margin% < 200 and we have an open trade, we might consider closing because margin call risk. But since we only open one trade at a time with small risk, margin level dropping below 200% likely means either account lost a lot (close to margin call) or other positions outside EA exist. We'll not auto-close our trade just for that (unless maybe margin < 100% which means margin call anyway). We will primarily use it to prevent new trade. - **Spread spike or volatility changes while in trade:** If spread unexpectedly jumps while in trade, we typically just let trade run to SL/TP. But if it's due to news event, our news filter should have prevented entry in first place. If news comes unexpectedly (unscheduled) and spread jumps massively, we might see slippage on SL. We can't do much mid-trade aside from perhaps closing early if environment changes drastically. - One could implement: if spread > X while trade open, close trade (to avoid huge slippage on SL). But that's an advanced precaution possibly not needed. Usually, if spread jumps, price likely spiked too hitting SL/TP quickly anyway. We'll skip that complexity now.

Expected Trade Outcome Distribution: - Most trades will either hit TP for +1.5R or SL for -1R within a few bars (maybe within 30-60 minutes). - Some trades might hover and get closed by time-out at maybe -0.5R or +0.5R or so (if they're in between). - Win rate should hopefully be around 50–60%. If our edge is good, maybe ~60%. That yields positive expectancy with 1.5R wins. - Loss streaks of 10 will be rare with a good edge and risk controls, but we have that covered if it happens (maybe in weird sideways or after big regime shift). - Profit factor hopefully >1.3 or so in tests which is decent.

Sessions and No-Trade Periods Enforcement: - The EA will simply not run signals outside 07-18 GMT. This means at 18:00, even if we have trend and signals, it won't enter new ones. - If a trade is open, it can continue until stopped or 3h or Friday cutoff. - Friday at 21:00, all trades closed, and no new ones of course after that until Monday. - Monday morning, we start fresh (the daily/weekly counters also reset daily or weekly accordingly). - News filter: if say FOMC at 18:00 GMT Wednesday, we will have stopped trading at 17:55 and resume at 18:15 (which is actually outside session, so effectively we resume next day). - This ensures we skip those known events entirely.

Everything above ensures the strategy **naturally stays within the desired trade frequency**: We are only checking once per M5 bar, in specific hours. That's 12 bars/hour * ~11 hours/day = ~132 bars/day to evaluate. The strategy might trigger maybe 0–2 trades on a typical day: - In a strong trend day, perhaps 1–2 pullback entries. - In a choppy day, perhaps 1 long and 1 short at two extremes. - Many quiet days will yield 0 trades (if no significant RSI extremes occur). - Big volatile swings might yield a couple trades if each swing gives a signal.

So over a week maybe a handful of trades, which fits ~100-300/year.

All the pieces (trend filter, RSI, pattern confirm, risk limits) work together to produce a robust scalping strategy: - It is **simple** (mostly using EMA, RSI, candle highs/lows), - **Frequent enough** but not over-trading, - Avoids the pitfalls that v1 had (like waiting for perfect liquidity sweeps only), - It will enter more often, e.g., any decent pullback in a trend rather than only when a stop-hunt occurs beyond a range.

This concludes the conceptual logic of v2.

Next step is to translate this into implementation tasks file by file, but as a summary:

Strategy pseudocode summary (for clarity, not actual code):

```
Every new M5 bar in session:
  if SpreadOK && not in no-trade period:
    Determine trend_mode and bias using EMA50 & ADX.
    if trend_mode and bias=LONG:
       if RSI<=30 and prior downmove and last close > prevHigh:
           signal = BUY with SL below recent low, TP at 1.5R.
    if trend_mode and bias=SHORT:
       if RSI>=70 and prior upmove and last close < prevLow:
           signal = SELL with SL above recent high, TP at 1.5R.
    if not trend_mode (ranging):
       if RSI<=30 and last close > prevHigh: (oversold bounce from low)
           signal = BUY (SL below low, TP 1.5R)
       else if RSI>=70 and last close < prevLow: (overbought drop from high)
           signal = SELL (SL above high, TP 1.5R)
    if signal exists:
       Compute lot for risk%, send order with SL/TP.
```

(This is a simplified representation. In implementation, we'll refine conditions and ensure we don't double-dip signals etc.)

By following this design, the EA will capture many of the scalping opportunities in gold: - Quick bounces after small dips in uptrends, - Quick reversals off intraday support/resistance in ranging conditions, - Always with protection and predefined profit target, - And it will **sit out** when conditions are not favorable (e.g., no trend and no extremes, or spreads too high, or after hitting loss limits).

We believe this strategy will hit the sweet spot described: around 100-300 trades/year, 5-60 min holding, low drawdowns, and resilience to volatility – essentially fulfilling all requirements.

(The design is ready to be coded according to these rules. Next, we map out how to modify each file of the v1 EA to implement this.)

8. System Architecture Migration Plan (File-by-File)

We will now go through each file in the repository and describe how it will be transformed or replaced for v2. For each file, we explain its current role (in v1) and the required changes for v2: - What logic to remove (v1-specific or now irrelevant parts), - What new logic to add (based on the v2 strategy above), - New inputs or outputs (parameters, return values, etc.), - How the file will interact with others in the new design.

This will serve as a blueprint for developers (or coding AIs) to implement v2 without needing context beyond this spec.

1. `XAUUSD-GOD.mq5` (Main EA file)
Current Role: The main expert advisor script that includes all modules and defines the event handlers (`OnInit`, `OnTick`, etc.). In v1, it set up M15 timeframe, built signals by calling `DetectRegime()` and

either `ScanAndSignal_LiquiditySweep()` or `...Breakout()`, and executed orders accordingly [45] [46]. It also handled gating (PreTradeGate) and logging.

Changes for v2: This file will be heavily modified to implement the new unified strategy: - **Timeframe**: Change timeframe usage from M15 to **M5**. In `OnInit`, set `tf = PERIOD_M5` and adjust any bar timing accordingly. For example, the global `g_lastBar_M15` can be renamed `g_lastBar_M5` and we use PERIOD_M5 in `NewBarGuard` [30] [21]. This ensures the EA triggers on new M5 bars. - **Session Input**: Ensure `Session_Start_Hour` and `Session_End_Hour` from config are actually used. In PreTradeGate (or CanTradeNow), incorporate a time check: only return true if current server hour is within [Start, End] and `Use_Session_Filter` is true. We may implement this either by expanding `CanTradeNow()` or adding a new function. Possibly easiest: have `CanTradeNow()` call a new `SessionOK(server_time)` that checks hour range and also if Friday beyond close time. - **BuildSignal logic**: Remove references to `Regime r = DetectRegime()` and separate liquidity vs breakout scanning [47] [48]. Instead, implement our unified entry logic: - Within `OnTick`, after PreTradeGate passes and `NewBarGuard` indicates a new bar, call a function (we can write inline or call `ScanForSignal()` we create) to evaluate entry conditions. - This function will incorporate what was described: reading indicators (EMA50, ADX14, RSI14, recent highs/lows) and setting up a Signal struct if criteria met. We might not use the old `Signal` type from v1 (though we can, as it holds fields valid/dir/entry/sl/tp). - Remove calls to `ScanAndSignal_LiquiditySweep()` and `ScanAndSignal_Breakout()` completely, as well as their includes, since we won't use those modules in v2. Instead, directly or via a new module, build the `Signal`. - E.g., `Signal sig; sig.valid=false; if(condition_long) { fill sig.dir=DIR_LONG, sig.entry=0 (market), compute sl & tp, sig.reason="SCALP"; sig.valid=true; } similarly for short.` - Possibly factor some code into a new file `entry_logic.mqh` to keep main neat (see below for new files). - **Order Execution**: Keep using `ExecuteMarketOrder()` as in v1 for immediate orders [49], but adapt conditions: - If `sig.valid`, compute lot via `ComputeLot(sig.entry, sig.sl)` [50] (that function is in risk.mqh). - Then call `ExecuteMarketOrder(sig, lot, ticket)` for both long or short signals (v1 separated by reason, but now reason could just be one type). - Remove `ExecutePendingOrder` usage (since no pending orders in v2 Option A). We might not need `Order_Expiration_Min` input anymore. Actually, we can leave it for future if needed but in Option A we won't use it. - Still respect `Max_Slippage_Points` (which is handled inside ExecuteMarketOrder via `SetDeviationInPoints`). - **Logging**: Update any log messages for clarity. For example, log regime decisions ("TREND mode LONG" or "RANGE mode SHORT signal" etc.), log why gate blocked (includes time, spread, etc.). We keep `LogEvent` usage for actions and `LogError` for errors. - **Integration with new/removed modules**: Exclude headers we no longer need: - Remove `#include <inc/liquidity_sweep.mqh>`, `<inc/breakout.mqh>`, `<inc/range_detector.mqh>` – these modules will be retired. Remove any constants or definitions related solely to them (like REASON_LIQ_SWEEP, REASON_TREND_BO from constants – we might replace with REASON_SCALP). - We will add (if we make one) `#include <inc/entry_logic.mqh>` for new unified signal generation. - **Ensure one-trade-at-time**: We might want to check in OnTick before building signal: if there's already an open position (`PositionsTotal()` with our magic > 0), skip building new signal. Alternatively ensure PreTradeGate or `CanTradeNow` covers it (we can implement `if(PositionSelect(_Symbol) && PositionGetInteger(POSITION_MAGIC)==MAGIC) then return false` in PreTradeGate). - This prevents stacking trades. - **PreTradeGate enhancements**: v1's PreTradeGate called `TM_DD_GateOK()` and `CanTradeNow()` [51]. We will: - Keep `TM_DD_GateOK` (daily DD check) and also add a weekly check function (we'll create `TM_Week_GateOK()` for weekly, similar style). - Expand `CanTradeNow` (in spread_vol_filter.mqh) to also check `SessionOK` and `noTradeUntil` (the pause after consecutive losses). - So PreTradeGate does: if(!TM_DD_GateOK() || !Week_GateOK() || !CanTradeNow()) return false. Also, if either daily or weekly gate fails, log a "GATE: Blocked due to drawdown" message as v1 did [52]. - If PreTradeGate returns false, OnTick logs reason (we'll include maybe "WeeklyDD= x%" if that’s the case). - **Magic number**: Ensure to maintain unique magic (v1 did in trade_manager or orders `SetExpertMagicNumber(MAGIC_BASE+Magic_Offset)` [38]). We'll reuse that mechanism; might not need changes except Magic_Offset default in config (1 is fine). - **Indicator initialization**: v1 `InitIndicators(sym,tf)` created ATR, ADX, EMA, RSI handles [53]. We will use all those indicators in v2. Possibly add handle for a second EMA (like if we want EMA200 as well) or Bollinger if we use it. But can avoid Bollinger by logic. Might consider adding a handle for a short SMA or something if needed. For now, existing ones suffice. - If we want a Bollinger or similar, we could either compute from available data or call iBands. Probably not needed as RSI covers it. - **Summary:** The main file will orchestrate the new strategy flow: - OnTick: check time/spread/gates, on new bar evaluate signal conditions, if signal then trade, etc. - Remove old regime switching complexities. The code becomes more straightforward sequential checks.

2. `inc/types.mqh`
Current Role: Defines custom types, particularly the `Signal` struct and `Direction` enum etc. In v1: - `enum Regime { REGIME_RANGE, REGIME_TREND };` - `enum Direction { DIR_LONG, DIR_SHORT, DIR_NONE };` - `struct Signal { bool valid; Direction dir; double entry, sl, tp; string reason; };`

Changes: We can reuse these for v2: - Keep `Direction` and `Signal`. They are useful to pass around signal info. - Possibly update the `reason` constants: - In constants.mqh v1 defined `REASON_LIQ_SWEEP` and `REASON_TREND_BO` [54] for logging. We can define a new one, e.g., `REASON_SCALP` or a couple like `REASON_TREND_PULLBACK` and `REASON_RANGE_REV` if we want to differentiate. - But not necessary to have multiple; could just tag all as "SCALP". - Add any new enums if needed (not likely). - Ensure `Signal.entry` usage: in v1, for market orders they often left `entry=0` which inside ExecuteMarketOrder used tick price instead [55]. For v2, since we're doing market immediate, we can similarly not worry about exact entry price (we’ll pass 0 or current Ask/Bid for reference). - Possibly add a field for expiry or time created if needed (not really needed). - So minimal changes: maybe rename `REASON_LIQ_SWEEP` to something else or repurpose `REASON_TREND_BO`. But better to add: - in constants: `#define REASON_SCALP "SCALP"` for logging/tracking. - Then when building signal do `sig.reason = REASON_SCALP`. - Other fields remain.

3. `inc/constants.mqh`
Current Role: Holds various constant definitions (EA version, symbol, magic base, min stop points, etc.) [56].

Changes: - Update `EA_VERSION` string to reflect new version (e.g., "XAUUSD-GOD 2.0"). - `SYMBOL_TARGET` and `TF_TARGET`: v1 had `TF_TARGET PERIOD_M15` [57]. Change `TF_TARGET` to `PERIOD_M5` (if used somewhere). This is used maybe in logs or checks. Actually, v1 not heavily used it beyond maybe OnInit logs. But update anyway. - Magic number: `MAGIC_BASE` can remain same (66050001). We can keep using Magic_Offset input to differentiate if user runs multiple EAs. - Add `REASON_SCALP` define as mentioned. - Possibly define any threshold constants default for filters: - e.g. `const int MAX_CONSEC_LOSS = 10;` but that we will take from config inputs likely. - `const int FLASH_CRASH_PIPS = 500;` if we want to define the threshold. - Many of these can be just in config inputs as parameters rather than constants. - We might remove `REASON_LIQ_SWEEP` and `REASON_TREND_BO` if no longer used. Or keep them unused, minor effect. Better remove to avoid confusion. - Keep `MIN_SL_POINTS` = 10 and `MIN_TP_POINTS` = 10 as v1 had [58]. Those are still useful to enforce not zero stops. We might adjust values if needed (10 points = $0.10, which is fine minimal). - If we require any new minimal constants (like min ATR or something), but that is in config as Min_ATR_Points.

4. `inc/config_inputs.mqh`
Current Role: Declares all user inputs with default values (grouped into categories) [59] [60] [61] etc.

Changes: We will revamp inputs to match v2 parameters, removing those related to old strategy: - **Core Indicators:** - Keep `EMA_Trend_Period` (default 200 in v1) but likely change default to 50 as per v2. We might actually want both a long and short EMA period (if we intend to use EMA200 in addition). Possibly add `EMA2_Period = 200` if we think of referencing a longer MA in logic. If not, just adjust the existing: - `input int EMA_Trend_Period = 50;` (from 200). - Remove `EMA_Intraday_Period` if not used (v1 had 50, but they never explicitly used that in code as far as I saw; maybe intended for something). - Keep `ADX_Period = 14;` and `ADX_Trend_Threshold = 25.0` (we might set default to 20.0 as per design). So: - `input double ADX_Trend_Threshold = 20.0;` - Keep `RSI_Period = 14;` (v1 had 14). - `RSI_Buy_Threshold = 35;` and `RSI_Sell_Threshold = 70;` (v1 had 35/70). We might adjust default to 30/70 or 30/70 symmetrical. Actually, to start maybe do 30 and 70 or 30/70. Or even allow different oversold/overbought? It's symmetrical instrument, so: - `input int RSI_Buy_Threshold = 30;` - `input int RSI_Sell_Threshold = 70;` - `RSI_Exit` (50) was in v1 (maybe intended to exit if RSI passes 50 baseline). We likely can remove `RSI_Exit` because we didn't plan a RSI-based exit, or keep if we want to possibly exit when RSI crosses back to 50 (some strategies do that as trailing exit). But currently, we stick to fixed TP/SL, so remove it to avoid confusion. - `ATR_Period = 14;` keep (for stops). - **Breakout Engine:** These can be removed entirely: - `Breakout_Lookback, Breakout_CloseBeyond_Points, Pending_Offset_Points, Breakout_Use_ADX_Filter, Breakout_Use_Volume_Filter, Vol_MA_Period, Vol_Min_Ratio` – none needed now since no separate breakout logic. - We remove this section to declutter or comment them out. Or if we think future v2.1 might reintroduce some breakout option, we can keep but set them not used. Better remove for clarity. - **Liquidity-Sweep Engine:** Also remove: - `Range_Lookback_Bars, Range_Min_Width_Points, Range_Max_Width_Points, Sweep_Buffer_Points, Sweep_Reentry_Confirm_Bars, Sweep_Use_RSI_Confirm`. All specific to v1 range logic, not needed in v2 (we have simpler conditions). - **Session Filters:** - Keep `Use_Session_Filter = true;` - Keep `Session_Start_Hour = 7;` and `Session_End_Hour = 18;` (back to original as comments say London open 7, but earlier code changed to 9? Actually in config they commented "BACK TO ORIGINAL (London open)" so I think 7 is correct). - Possibly add `Friday_Close_Hour = 21;` or similar if we want that adjustable. v1 has `Friday_CloseTime = "21:00";` as string [62]. They used string which is not ideal (maybe for easier comparison). - We could use an int hour for simplicity (21). - But since they already had string, maybe they planned to parse it. Simpler: change `Friday_CloseTime` to int (21) or at least specify clearly it's 21:00. Actually, we can keep it as string and parse in SessionOK function by using `utils.mqh IsTimeWithin` or direct compare of hours/mins. - I'll keep `input string Friday_CloseTime = "21:00";` as in v1. We'll implement accordingly. - **Risk/Sizing:** - `Risk_Mode` (0 for fixed lot, 1 for percent) in v1. We can keep to allow fixed lot mode. But usually percent risk is better. We'll keep it for flexibility: - Default `Risk_Mode = 1` (meaning percent mode, maybe set 1 as code for percent because v1's logic was if(Risk_Mode==0) use fixed lot, else use percent [63]). - `Risk_Percent = 0.5;` (change from 2.0 to 0.5 default). - `Fixed_Lot = 0.01;` (same). - `Max_Daily_Drawdown_Percent = 3.0;` (down from 5.0). - **Add** `Max_Weekly_Drawdown_Percent = 7.0;` (new input). - Could add `Consecutive_Loss_Limit = 10;` (int) to set the threshold of losses in a row. - Possibly add `Loss_Pause_Duration_Min = 120;` for 2 hours, if we want that configurable (though it's mandated, making it input allows tweaks). - We should also consider risk scaling if equity grows/drops (we are using percent risk so auto-adjusts). - **Stops/Targets:** - `ATR_Mult_SL_Trend` and `ATR_Mult_SL_Range` in v1 were 1.50 and 1.20. We might not use separate values if we unify stops by structure + ATR. But we could repurpose them: - Maybe use `ATR_Mult_SL = 1.3;` as single input if we choose to use ATR for all. - Or keep two if we indeed decide trending trades use slightly larger stops than ranging. That was idea in v1, might not need. To be safe, maybe keep them but adjust defaults: - `ATR_Mult_SL_Trend = 1.3;` - `ATR_Mult_SL_Range = 1.0;` - These can be used such that if ADX high use trend one, if ADX low use range one. We can implement that in our SL calc. - Remove the partial trailing ones from v1: - Actually trailing and partial are in Management section below. In this stops section, they only had ATR multipliers for SL. - They also had no explicit TP multiplier except in Profit Targets as RR_Target. - **Execution Gates (new name maybe Spread/Vol Filter):** - v1 had `Min_ATR_Points = 100;` and `Max_Spread_Points = 1500;` [64]. - Change `Max_Spread_Points` default to `50;` as per our constraint ($0.50). - Min_ATR_Points = 100 is fine (meaning $1). - We keep them to enforce volatility/spread gating. - **Management (Trailing/Partial):** - v1 had `Trail_Mode = 2; Trail_Start_Points=150; Trail_Step_Points=120; Trail_ATR_Mult=1.00; Move_BE_After_Points=150; Use_Partials=false; Partial1_Ratio=0.50; Partial1_Target_Points=200;`. - For v2: - Default `Trail_Mode = 0;` (disabled trailing by default since Option A). - Keep others for potential Option B usage but not active by default. We'll note: - `Use_Partials = false;` (remain false). - Partial1_Ratio 0.5, Partial1_Target_Points whatever (maybe 100 or 150 like was). - Actually v1 Partial1_Target_Points=200 (2 dollars or 200 pips? Maybe). - We can adjust to align with our typical R: If risk ~100 points, 150 points is ~1.5R. But anyway, not used unless user turns on partial. - It's fine to keep them for future or advanced users. - If we think clutter, we could hide them by grouping or commenting "for future use". - **Profit Targets:** - v1 had `TP_Mode = 1; TP_Fixed_Points = 200; RR_Target = 2.0;`. - We will use RR mode (1) by default, and RR_Target = 1.5 (change from 2.0 to 1.5 as recommended). - TP_Fixed_Points not used if in RR mode, but we can keep it if user chooses fixed pips mode. - Possibly update comment from "Target 2:1 RR" to "Target 1.5:1 RR". - **Housekeeping:** - Keep `Magic_Offset, Max_Slippage_Points, Order_Expiration_Min, Order_Comment`. - `Max_Slippage_Points = 50;` if not already (v1 had 50). - `Order_Expiration_Min = 30;` (not used in v2 unless pending, but keep). - `Order_Comment = "XAUUSD-GOD-M15";` - change to "XAUUSD-GOD-M5" or just "XAUUSD-GOD". - Actually, maybe change to reflect version as well like "XAUUSD-GOD-v2". - **Trading Schedule (Days):** - v1 had booleans for Mon-Fri (all true) and `Friday_CloseTime = "21:00";`. - We'll keep those. We will use Friday_CloseTime to enforce Friday closure as said. - Possibly add `Pause_Between_Trades_Min = 0;` or specifically use the consecutive loss logic instead, so maybe not needed. - **Misc:** - `Only_New_Bar = true;` (keep to ensure we only trade once per bar). - `Order_Type = 0;` (likely leftover from risk mode logic or maybe for switching pending/market in v1). - We can interpret 0 as auto (market or pending based on reason). - We won't really use it, as we will always do market orders in v2 for simplicity. So this input could be removed or left in case user wants to force pending? Better to remove to not confuse.

Summarizing, config_inputs will have fewer entries after removing breakout/liquidity ones. We will ensure new ones for weekly loss etc. are added.

5. `inc/utils.mqh`
Current Role: Utility functions for time and price formatting (DigitsFor, PointFor, NormalizePrice, IsTimeWithin, IsNewBar, LatestTick) [65] [66] [67].

Changes: - Likely keep everything and add a couple new helpers: - A `SessionOK(datetime t)` function: returns true if `Use_Session_Filter` is false (means no filter) or if true and `Session_Start_Hour <= t.hour < Session_End_Hour` and day is Monday-Thursday or Friday before close time. - For Friday, also check if t.hour < friday_close_hour (21). If it's equal or later, then session not OK (we stop trading). - Actually, our trading days booleans (Trading_Day_Fri etc.) could cover not trading after Friday close if we set Trading_Day_Fri = false after 21:00 via DayAllowed? But DayAllowed in filters currently only works by day, not time. - So implementing it in SessionOK is cleaner. - Alternatively, modify `DayAllowed` or incorporate time to `CanTradeNow`. - We have an `IsTimeWithin(string hhmm, datetime)` in utils which can check exact minute match. Not exactly needed for our use. - Possibly an `IsFridayCloseApproaching(datetime)` function or so, but not necessary if we incorporate in SessionOK or directly in PreTradeGate. - `IsNewBar` is already used to detect new timeframe bars given last stored time [67]. We will keep using that (g_lastBar_M5 etc.). - Possibly add a function to parse the Friday_CloseTime string to hours/min if needed. But if it's "21:00", we can just compare with dt.hour and dt.min easily by splitting string or using that IsTimeWithin. - Actually, we could do: `if(IsTimeWithin(Friday_CloseTime, now))` to detect exactly 21:00. But we need >=21:00. - Might be simpler: in code, convert Friday_CloseTime to an integer hour (21) by `StringSubstr` etc., or require user input as "21:00". - Or we ignore the input and just hardcode 21:00 since requirement said 21:00. - But let's use it: parse hour = (int)StringToInteger(StringSubstr(Friday_CloseTime,0,2)), minute = (int)StringToInteger(StringSubstr(Friday_CloseTime,3,2)). Then in SessionOK, if dt.day_of_week==5 (Fri) and dt.hour*100+dt.min* >= *hour*100+min, then false. - Something along those lines. - Summation: Add `bool SessionOK(datetime)` possibly to make code clean. Otherwise we can do inline in CanTradeNow with dt struct.

6. `inc/indicators.mqh`
Current Role: Manages the indicator handles and provides functions ATR(bar_index), ADX(bar_index), EMA(bar_index), RSI(bar_index), VolumeMA(bar_index) [53] [68].

Changes: - Since we changed timeframe to M5, but they pass tf from OnInit to InitIndicators, which sets `g_tf = tf`. We will pass PERIOD_M5 now. The indicator handles will then be M5. - We might add additional handles: - If we want an EMA200 in addition to EMA50 for long-term context, we could create `g_handleEMA2 = iMA(symbol, tf, EMA_Long_Period, ...)` in InitIndicators if we have such input (like if we keep EMA_Trend_Period = 50 and we want maybe EMA_Long_Period=200). - If we don't explicitly need it, skip. Possibly not needed for logic; we can rely just on one EMA or trend conditions. - If not adding any, the existing ATR, ADX, EMA, RSI are sufficient. Actually EMA was 200, we changed to 50 by input, so fine. - Might adjust VolumeMA if volume filter used. But probably not needed if we skip volume filter. - So minimal changes. The fixed issues (like proper ArraySetAsSeries calls in v1 code) are already done in this file, no need to change. - If adding second EMA: - Add static int g_handleEMA2. - In InitIndicators: `g_handleEMA2 = iMA(symbol, tf, EMA_Long_Period, 0, MODE_EMA, PRICE_CLOSE);` and check not INVALID_HANDLE. - Provide a function if needed to get it, e.g., double EMA2(int index) similar to EMA. - Only do if we plan on using it (maybe not). - We'll probably not use second EMA at first to avoid confusion. If need something like 200 EMA used to identify major trend or support, we can integrate later. For now, one EMA50 plus ADX covers trend identification moderately.

7. `inc/regime.mqh`
Current Role: Provided `DetectRegime()` by comparing ADX to threshold (25) [69].

Changes: - This function becomes less central because we won't classify regime in the same separate way. We have integrated logic: - Instead of returning a binary regime used to branch to separate logic, we directly incorporate ADX logic in entry conditions. - We can repurpose or remove `DetectRegime`: - Might still use it to know if trending vs ranging: It returns REGIME_TREND if ADX>=threshold else REGIME_RANGE. - We could still call this at start of OnTick to set a flag (ranging or trending). - But since we might also need to know direction (which DetectRegime doesn't give, only says trend or range), we anyway have to check price vs EMA for bias. - So `DetectRegime` could be used in combination with an EMA check to decide `trend_mode` in our logic. - It's a one-liner basically, we could just do ADX(1)>= threshold in main code. - It's fine to keep the function, maybe rename Regime to Mode if we want. But not necessary. - Minimal: update threshold usage if we changed ADX_Trend_Threshold default to 20. It's referencing `ADX_Trend_Threshold` input already [39], so fine. - We might decide to remove the whole Regime enum to avoid confusion with v1's usage since we won't branch to two separate function calls by regime. However, could keep to reuse notion: - e.g., if(DetectRegime()==REGIME_TREND) then trend_mode=true else false. - It's okay to keep for clarity. It's a quick call to ADX. - On balancing, we keep it for easy reading.

8. `inc/filters.mqh`
Current Role: Contains filter functions SpreadOK, ATRSane, DayAllowed, RSIConfirm, VolumeConfirm [70] [71] [72].

Changes: - **SpreadOK:** Currently returns true if spread_points <= Max_Spread_Points [73]. We'll keep that logic, just ensuring our Max_Spread_Points default is now 50. Possibly add an additional condition: if `Max_Spread_Points<=0` then always true (they had that). - It's fine as is. Might add debug log if false to know current spread but not needed here (we log in PreTradeGate). - **ATRSane:** Returns true if ATR >= Min_ATR_Points [74]. Keep, as we plan to use to avoid ultra quiet times. - If `Min_ATR_Points <=0` then always true (they had that check). - Could consider different ATR threshold for different conditions but not needed. Keep one. - **DayAllowed:** They implemented allowed days Mon-Fri using Trading_Day_ *booleans [25]. Keep as is. If a user sets a day false, EA won't trade that day. - We might incorporate a further check for Friday after a certain time here: But since DayAllowed is only by day_of_week, better to handle time in SessionOK. - We'll keep DayAllowed for daily enabling/disabling. If needed, the user could set Trading_Day_Fri=false as a way to not trade Friday at all. - But since we do trade Friday until a cutoff, we leave Friday true and handle cutoff in Session filter/time. - RSIConfirm: In v1, used to confirm trade direction by RSI threshold (for Sweep_Use_RSI_Confirm option) [75]. We might reuse or repurpose this: - It returns true if RSI is beyond threshold for given direction: for long, r <= RSI_Buy_Threshold, for short, r >= RSI_Sell_Threshold [76]. - This is essentially exactly what we need to check for our entry signals. We could use this function to avoid duplicate code: - If we have Direction d, RSIConfirm(d) returns whether RSI is extreme in that direction. - Alternatively, we may need to incorporate it more fluidly (like if in trend mode maybe threshold different). But base function uses static thresholds (which we set as 30/70). - We can still use RSIConfirm to check oversold/overbought easily. - So keep it. Possibly adjust: - It currently returns true if threshold <=0 then always true (meaning if user sets threshold 0 to disable confirm). That might not be used often but nice optional. - We'll ensure our logic calls RSIConfirm at appropriate times. - VolumeConfirm:** In v1, used tick volume vs moving average [77]. Since we are not focusing on volume filtering for now (we rely on ATR/volatility instead), we can either leave it but not use, or remove it to simplify. - It was tied to Breakout_Use_Volume_Filter in v1 (if false they didn't call it). We removed those inputs likely. - We might not call VolumeConfirm anywhere in v2. So we could remove or simply leave it (it doesn't harm). - Perhaps remove to not confuse since we won't set Vol_MA_Period normally. - Possibly add new filter function: - E.g., `bool SessionAllowed(const datetime t)` which checks both DayAllowed and within hours (but we said we'll do hours in spread_vol_filter's CanTradeNow). - Or incorporate time in CanTradeNow directly (likely do that).

9. `inc/spread_vol_filter.mqh`
Current Role: Provided top-level `CanTradeNow(symbol, time)` that returned true if SpreadOK && ATRSane && DayAllowed [22], and `NewBarGuard(tf,last_time)` to enforce Only_New_Bar setting [78].

Changes: - **CanTradeNow:** We will extend it to include our new gating conditions: - Check `if(!SpreadOK(symbol)) return false;` - Check `if(!ATRSane()) return false;` - Check `if(!DayAllowed(server_time)) return false;` - **Add**: `if(Use_Session_Filter && (server_time hour not in [Session_Start, Session_End] or if it's Friday after Friday_CloseTime)) return false;` - We will need to get server_time's hour/min via MqlDateTime. - Possibly implement that logic here directly or via a SessionAllowed util. - **Add**: If we have a global `noTradeUntil` timestamp (set when pause after losses), check `if(server_time < noTradeUntil) return false;`. - `noTradeUntil` can be a static datetime defined perhaps in trade_manager or main. We could manage it in trade_manager or at least make it accessible here (maybe as an extern or we pass it in as we do server_time). - Simpler: since PreTradeGate in OnTick has `server_time` and we can access a global for noTradeUntil, we can incorporate it here if we include trade_manager or so. Or set a static in this file. Could define `static datetime g_noTradeUntil = 0;` and have a function to set it when needed (or just set from outside). - Perhaps better: we will handle pausing in trade_manager's ManageOpenPositions or after trade closed logic, which sets a global or static. Then check it here. - Check if margin level is okay: We can get margin level via `AccountInfoDouble(ACCOUNT_MARGIN_LEVEL)`. If < MinimumMarginLevel% (200) and `MinimumMarginLevel > 0`, then return false. - We might set `input double Minimum_Margin_Level = 200;` if we want user adjustable. Or just hardcode 200%. For flexibility, let's add in config as maybe "Min_Margin_Level_Percent = 200.0;". - Then in CanTradeNow: if Min_Margin_Level_Percent > 0 and margin_level < that, then block. - If blocked, we should log something. But logging is done in PreTradeGate, which logs if either TM_DD or Spread blocked (they appended reason string). We can extend that to log "MarginLow" if this fails. - But implementing logging within CanTradeNow might break pattern. Instead, PreTradeGate's logging currently does: - If !TM_DD, add "DD=x% " to reason. - Always adds "Spread=y ATR=z" to reason for block (v1 did). - We can similarly modify PreTradeGate: - It calls CanTradeNow and if false, it doesn't know which subcondition failed unless we replicate logic or modify CanTradeNow to set some global flags. Simpler: we can in PreTradeGate build reason by calling SpreadOK and ATRSane individually to see what failed (like v1 did). - Actually v1 did: after PreTradeGate false, they built reason string: - It checked if !TM_DD_GateOK then reason += "DD=..%". - Then reason += "Spread=val", and if ATR exists reason += " ATR=val". (It didn't say "ATR too low", it just logged current ATR). - It didn't explicitly say if ATRSane was false. But by including ATR value, user can infer if ATR was below min because they'd know threshold 100 vs reported value. - Maybe we should explicitly mention "ATR low" if ATRSane is false. - For margin or session filter, we should indicate in logs if blocked due to those: - e.g., reason text could include "SessionOff" or "Margin=123%" etc. - Implementation: in PreTradeGate, we can do: - if(!TM_DD_GateOK()) reason += "DD="+DoubleToString(curr_dd)+"% "; - int spread = SymbolInfoInteger(_Symbol,SYMBOL_SPREAD); - double atr = ATR(1); - reason += "Spread="+spread; - if(atr != EMPTY_VALUE) reason += " ATR="+DoubleToString(atr,2); - Additionally, if `Use_Session_Filter` and current time out of range, reason += " SessionOff"; - If margin low, reason += " Margin="+IntegerToString((int)margin_level)+"%"; - If consecutiveLossPause active, reason += " Paused"; - We gather these flags by checking conditions again or maintaining global states. It's fine to recompute here as overhead is tiny. - This way, the log line might look like "Blocked: Spread=55 ATR=0.8 SessionOff" if session ended, or "Blocked: DD=3.2% Spread=20 ATR=1.5 Paused". - We'll implement accordingly. - So CanTradeNow remains just boolean logic no logging, PreTradeGate will handle logging the reasons comprehensively. - **NewBarGuard:** Keep as is. It uses Only_New_Bar input and IsNewBar function. That's working as needed for per-bar logic. Only change is ensure it uses correct timeframe constant (we pass

PERIOD_M5 to it). - Remove any call to volume filter (VolumeConfirm) because we won't call it in PreTradeGate or anywhere for now.

10. `inc/risk.mqh`
Current Role: Functions for risk management: - SymbolVolumeSpecs, PointValuePerLot, ComputeLot (which handles fixed vs percent risk calculation) [79] [80] [43].

Changes: - We mostly keep these as is because they already do what we need (ComputeLot). - Check that `Risk_Mode` logic suits us: v1 did if(Risk_Mode==0) fixed lot, else percent [63]. Good. - We will change default Risk_Mode to 1 so it uses percent risk by default. - Also check that they use Risk_Percent and Fixed_Lot from inputs (yes they do). - Ensure `Max_Daily_Drawdown_Percent` usage in trade_manager for gate (they already had). - Possibly add Weekly DD in a similar manner: - In trade_manager, we will add a new static or global to track weekly data and a function like `bool TM_Week_GateOK()` similar to TM_DD_GateOK but on week. - Actually risk.mqh doesn’t handle drawdowns, that was trade_manager. - So risk.mqh might not need changes. - Perhaps add any function if needed for computing something else (like risk to reward?), but likely not.

11. `inc/orders.mqh`
Current Role: Wraps the MT5 CTrade functions to execute orders: - `EnsureTradeInit()` sets magic and slippage [81]. - `ExecuteMarketOrder()`: uses CTrade.Buy or Sell to execute at market, sets SL, TP, comment [82] [83]. - `ExecutePendingOrder()`: for placing BuyStop/SellStop with expiration [84] [85]. - `ModifySLTP()`: to modify stops (used by trailing) [86] [87]. - Also some internal checks like RespectMinStopDistance which ensure order’s SL/TP are beyond min stop level (brokers have min distances) [88] [89].

Changes: - We will still use `ExecuteMarketOrder` for all our entries. - No changes needed except maybe the comment: - It builds comment as `Order_Comment + "|" + reason` [90]. Our Order_Comment default was "XAUUSD-GOD-M15"; we should change to reflect M5 or v2. - Also `reason` we set as "SCALP" or similar. So an order comment might be "XAUUSD-GOD-M5|SCALP". This is fine. - Remove or keep `ExecutePendingOrder`: We won't use it in v2, but might not harm to keep if in future needed. If we want to declutter, we could comment it out or leave since it's isolated. - Check `Max_Slippage_Points`: In EnsureTradeInit, they set deviation = Max_Slippage_Points [29]. Our default is 50 which is fine. If user changes it, it’s used. - No other changes likely needed. It's stable.

12. `inc/trade_manager.mqh`
Current Role: This manages open positions: - It tracks daily drawdown (g_dd_day, g_dd_start_equity, g_dd_peak_equity) and updates each tick via TM_DD_Update() [91]. - Provides TM_DD_GateOK() to enforce Max_Daily_Drawdown_Percent [16]. - ManageOpenPositions() loops through positions and applies break-even, trailing, partial closes using the helper functions ApplyBreakEven, ApplyTrailing, ApplyPartial [92] [93].

Changes: - **Daily & Weekly tracking:** - They used g_dd_day to detect new day by comparing day != current day to reset daily stats [94]. - We need similar for week. Possibly maintain `static int g_dd_week = -1; static double g_week_start_equity=0; static double g_week_peak_equity=0;`. - On TM_DD_Update (which runs every tick), we also check if new week: - e.g., if(current_date.week_of_year != stored week or if current day_of_week < stored day_of_week meaning a wrap) then reset weekly. - Simpler: if current is Monday and g_dd_week != Monday's date or something, then reset. - Actually maybe easier: if day_of_week < g_dd_day (means we've moved from Sunday to Monday likely) then new week, or track ISO week number. - For robust: Use `MqlDateTime dt; TimeToStruct(tv, dt);` and check `if(dt.day_of_week == 1 && g_dd_week != dt.day)`, meaning Monday and last stored wasn't this Monday. - Actually in code, day_of_week uses 1=Monday, etc up to 5=Fri, 6=Sat, 0=Sun for MT5, likely. If at tick time dt.day_of_week == 1 (Mon) and (g_dd_week != dt.yyyy*100+dt.mi (month) maybe, something unique like year+week number). - Could compute week number by some formula but complicated. - Perhaps easiest: if dt.day_of_week == 1 (Mon) and g_dd_day_of_week last update was > 1 (meaning last tick was Sun or Sat?), but EA wouldn't tick on weekend as no ticks. Or if it's Monday and g_week_start_equity is 0 or g_dd_week stored was from different month etc. Actually, we can do more straightforward: store last reset date, if current date >= last reset date + 7 days then reset. But markets off weekends so not continuous. - Another approach: If we do daily resets, we can accumulate daily beyond day resets to track weekly. But simpler: attempt to implement as: Each day in TM_DD_Update, if NewServerDay or g_dd_day < 0, we do daily reset and also if it's Monday do weekly reset. - That covers normal cases (we assume user runs EA continuously through weeks). - So:*

```
if(NewServerDay(tv) || g_dd_day < 0) {
    DD_Reset(tv);
    if(dt.day_of_week == 1) {
        g_week_start_equity = eq; g_week_peak_equity = eq;
    }
}
```

where eq = AccountEquity and dt is current's date. - We should store something to identify if week changed in case user starts mid-week, but if they start mid-week, we set those anyway on first call. - We'll also need to update g_week_peak_equity daily: If eq > g_week_peak_equity then g_week_peak_equity = eq, similar to daily but for week. - Then implement `double WeekDD_CurrentPercent()` *to compute (g_week_peak_equity - current equity)/ g_week_peak_equity * 100. - Then* `bool TM_Week_GateOK()` *returning (weekDD < Max_Weekly_Drawdown_Percent). - We'll call TM_Week_GateOK in PreTradeGate similarly to TM_DD_GateOK. - Might also track g_dd_week as int to know if we reset already for the week, but not needed if above logic resets on Monday. Edge: what if Monday is holiday or no ticks until Tuesday? Then our code wouldn't reset on Monday because no tick, it would reset on first tick Tuesday but the condition dt.day_of_week==1 fails because it's 2. So it wouldn't reset at all that week - problem. - In that case, if market closed Monday (rare except maybe Xmas?), we should still consider Tuesday’s first tick as new week. But our code wouldn't catch because it's not Monday. - We might use an alternative check: if current day_of_week < previous day_of_week (meaning we rolled over weekend) and previous day_of_week was Friday or Saturday or Sunday -> then it's a new week if we see day decreased. - Actually, we track g_dd_day (which is day of month) for daily. For weekly we could track last known week number if we can get it. - Another simpler approach: Use day_of_year: if current day_of_year < last trade day (meaning year rolled? too far). - Or use total days count since some epoch for Monday midday and add 7. - Given complexity, maybe acceptable that if Monday had no ticks and EA didn't reset weekly, it might erroneously carry last week’s peak. That scenario extremely rare (market always opens Monday except special occasions, and even then Monday night open is Monday, so will get tick Monday). - We'll proceed with Monday check with the understanding it's 99% fine. - TM_DD_Update: - Already updates daily. We integrate weekly as above. - TM_DD_GateOK: - Already uses Max_Daily_Drawdown_Percent. We'll add similar* `TM_Week_GateOK()` *to compare to Max_Weekly_Drawdown_Percent. - In PreTradeGate, we'll call both (with an OR logic: if either daily or weekly not OK then gate fails). - Consecutive loss pause*: - trade_manager currently doesn't handle anything with consecutive losses. - We need to detect trade results. Possibly in ManageOpenPositions or after a trade closes. But trade_manager has no explicit hook for closure events in this design (since OnTick just calls TM_ManageOpenPositions to manage trailing etc., not to check closed trades). - We might implement this in TM_ManageOpenPositions as well, but that function loops open positions only; it doesn't handle closed ones. - Better approach: Use OnTick to detect closed trades via history or a global variable: - We can track number of open positions last tick vs now. If it was 1 and now 0, a trade closed. - Or check if `PositionsTotal()==0` and previously we stored a flag that a position was open. - Alternatively, after sending an order we set a flag or store its ticket. Then in OnTick, check if that ticket still open. - For simplicity, possibly implement in ManageOpenPositions: when it loops through positions, if it finds none and maybe had one before, but it has no memory of before. - Could make g_last_position_count static in this file, update it each tick, and if it decreased and previous was > now, then a trade closed. Then we can query last closed trade's result. - Or simpler: Use History: Myfxbook signals would use trade events, but in EA, we can do: - If no positions open and we had one open before, then likely just closed (unless user closed manually or SL/TP triggered). - We can then use `HistorySelect` to get last closed deal for our symbol & magic to see if profit>0 or <0. - HistorySelect(TimeCurrent()-..., TimeCurrent()) to cover recent interval and HistoryDealsTotal to find last one. - Or maintain our own counters for wins/losses whenever we close in code. But since closure might be by server (stop hit), we should catch via history. - We might implement a function `CheckRecentClosedTradeOutcome()` called if we detect a closure. - However, easier: We know exactly when we close trades manually: - We close on TP or SL triggered by broker but we might not intercept exactly when. But trailing and partial we handle, however TP/SL by broker we get no direct code trigger. - Actually, MT5 has `OnTradeTransaction` event that can catch deals/closes, but that's more complex and not used in this code. - We'll do a simpler approach: - Add in trade_manager: static int last_loss_streak (to track), static bool position_was_open = false. - Each OnTick (TM_ManageOpenPositions is called every tick): - At top of TM_ManageOpenPositions, store int count = number of our positions (with our magic). - If count > 0, set position_was_open=true. - If count == 0 and position_was_open==true: - This means a position that was open is now closed (since it was open on previous tick). - So we then determine if the closed trade was win or loss: - Could track profit from global or via history. Possibly simplest: use `AccountInfoDouble(ACCOUNT_PROFIT)` difference but that includes all currently open maybe. - Better: query trading history. - We can retrieve closed positions in PositionsHistory, but easier to use HistoryDeal as deals include closures. - Use `HistorySelect` from (TimeCurrent()-some_delta) to TimeCurrent() and loop through deals for our symbol and magic looking for DEAL_ENTRY_OUT. - Or use trade manager partial knowledge: When we modify trailing or partial we log events but not final closure. - Might not be worth heavy coding; simpler: - We know if trade was profit or loss by comparing entry and exit price and direction, or by checking profit. - Actually easier: just recalc profit = if we had recorded entry price and lots, but if partial closed maybe multiple deals. - Actually, maybe simpler: maintain our own global for last trade outcome: - When we send a trade (in ExecuteMarketOrder), we can store the expected SL and TP outcomes: for instance, store a boolean lastTradeWasLong, and double lastTradeSL, lastTradeTP, and volume. - Then on closure detection, check current price relative to those: if price hit TP or exceeded in favorable direction, likely a win; if hit SL or beyond adverse, a loss. - But price might have moved further after closure, not reliable. - Or better, after closure, the account equity changed. But if other trades or manual changes, not reliable alone. - Honestly, using trade history is more robust albeit a bit of work. Let's do that: - On closure detection:

```
HistorySelect(TimeCurrent()-86400, TimeCurrent()); // select last 24h deals
int deals = HistoryDealsTotal();
for(int i=deals-1; i>=0; --i) {
    ulong deal_ticket = HistoryDealGetTicket(i);
    if(HistoryDealGetInteger(deal_ticket, DEAL_MAGIC) ==
(MAGIC_BASE+Magic_Offset) && HistoryDealGetInteger(deal_ticket, DEAL_ENTRY) ==
DEAL_ENTRY_OUT) {
        double profit = HistoryDealGetDouble(deal_ticket, DEAL_PROFIT);
        if(profit >= 0.0001) { last_loss_streak = 0; }
        else if(profit <= -0.0001) { last_loss_streak++; }
        else { // near zero
             // if it's exactly 0, count as break-even (reset streak maybe or
increment? probably reset because it's not a loss).
             last_loss_streak = 0;
        }
        break;
    }
}
```

- This finds the most recent closing deal for our EA. Profit includes commission? Actually DEAL_PROFIT should be net profit including commissions. - We consider a tiny profit/loss as per threshold (0.0001 to avoid floating precision issues). - If found, update `last_loss_streak` accordingly. - Then set position_was_open=false (ready for next). - Use `g_noTradeUntil`: if last_loss_streak >= Consecutive_Loss_Limit (10), then set `g_noTradeUntil = TimeCurrent() + Loss_Pause_Duration_Min*60;` and maybe reset last_loss_streak=0 or let it continue? Probably reset to 0 after pausing, because after break we treat new streak fresh. - Also log something like "Max loss streak reached, pausing 2h". - We have to include `Trade/Trade.mqh` to use CTrade, but history functions are MT5 global so no additional include needed (they are from `Trade.mqh` or standard). Actually, `HistorySelect` and `HistoryDealGet...` are standard functions, no extra include needed beyond `<Trade/Trade.mqh>` which we have anyway. - Efficiency: deals loop at most all deals of one day which is fine (only we trade at most few times day). - This approach ensures we catch both SL, TP, partial outcomes. The last closing deal might be partial or final closure; but since we always close full position at once in v2 (no partial by default), one DEAL_ENTRY_OUT should represent the trade closure with full profit/loss. - If partials/trailing in future, we may adjust logic to accumulate. - So incorporate above in ManageOpenPositions or in PreTradeGate (but better in Manage because it's tick-level and after trailing logic). - Actually, ManageOpenPositions is called every tick as well, so it's fine. After it finishes trailing adjustments, we can do closure detection at end of it. - Always update position_was_open = (count > 0) for next tick. - last_loss_streak likely static above ManageOpenPositions to persist across ticks.

- **Trailing/Break-even/Partial**:
    - They exist as functions (ApplyBreakEven, ApplyTrailing, ApplyPartial) and are invoked for each position each tick [92] [93]:
    - If (profit_pts >= Move_BE_After_Points) -> move SL to BE, etc.
    - If trailing mode do as configured.
    - If partial config triggers, do partial close.
    - By default we set trailing/partials off, so these if conditions won't execute (because Use_Partials=false, Trail_Mode=0).
    - But the code will still loop and check them each tick on open trades. That overhead is fine with one trade.
    - We can leave them, so advanced users could enable trailing by input if desired.
    - If we want to hard-enforce Option A and not allow partial/trail, we could comment out those calls or set conditions to false explicitly.

    - Better is leave for flexibility but default disabled. We'll document recommended Option A but allow turning on Option B features.
    - So keep code, but ensure defaults as said (Trail_Mode=0, Use_Partials=false).
- **Margin level tracking**: not explicitly needed here, we handle margin gating in CanTradeNow.
- **Other interactions**:
    - Ensure to include this file in main, it is included in XAUUSD-GOD.mq5 at end with orders.
    - We'll add references in trade_manager for new static (g_noTradeUntil, if we decide to store it here or in filters).
    - Actually, maybe define `static datetime g_noTradeUntil` at top of trade_manager (since trade_manager runs logic each tick).
    - trade_manager can also provide a function to set the pause. Or we just set it inside ManageOpenPositions when needed.
    - But we need to check it in CanTradeNow (in spread_vol_filter). Possibly easiest: make g_noTradeUntil a global extern in spread_vol_filter that trade_manager can set.
    - Or put g_noTradeUntil in spread_vol_filter as static and have a function in trade_manager to update it (like a setter).
    - Simpler: place g_noTradeUntil static in trade_manager, and in CanTradeNow reference trade_manager's variable via `extern datetime g_noTradeUntil;`.
    - That requires trade_manager to come before spread_vol_filter in includes or ensure extern is declared properly.
    - Alternatively, define g_noTradeUntil in a common place (like at top of main file as global static).
    - Considering architecture, probably easiest: define `static datetime g_noTradeUntil = 0;` at top of XAUUSD-GOD.mq5 (so it's global in that compilation unit).
    - Then in spread_vol_filter CanTradeNow, refer to ::g_noTradeUntil (should be visible if declared above include).
    - And in trade_manager when needing to set it, do `g_noTradeUntil = TimeCurrent() + Loss_Pause*60;`.
    - That might be simplest because main includes trade_manager and filter after, so it should know that global static.
    - Let's do that: in main .mq5, after includes, declare `datetime g_noTradeUntil = 0;`.
    - Then we don't need extern since it's in same compile unit.
    - We'll manage its logic in trade_manager and check in CanTradeNow.
    - The `Trade Wizard team` aside, we must be careful with linking. But since they all included in one program, the static at global scope is fine.

So those cover all files. Additionally, since we remove inc/liquidity_sweep.mqh, inc/range_detector.mqh, inc/breakout.mqh from build: - We should delete or exclude them from project to avoid confusion. But at least not include in main. - If any logic from them needed? range_detector had BuildRangeBox, not needed; breakout had some pattern logic but we wrote our own simpler. - We'll drop them.

That covers the file modifications needed for migrating to v2.

9. Final Output: Task List for Other AIs

Now, we provide a structured list of implementation tasks for transitioning the codebase from v1 to v2. Each task specifies which file/function to rewrite or update, what the new logic is, what inputs/outputs it deals with, and any edge cases or references to design constraints above. This list acts as a to-do list for developers (or coding AI agents):

Task A – Update Main EA (XAUUSD-GOD.mq5) for v2 Logic
Description: Rewrite the main expert advisor flow to implement the new scalping strategy on M5 timeframe. Remove old regime branching and integrate unified signal generation and trade execution.
- **Inputs:** Uses global inputs from `config_inputs.mqh` (session hours, risk settings, thresholds, etc.). Accepts tick data via `OnTick()` and time via `TimeCurrent()`.
- **Outputs:** Places trade orders (market orders) via `ExecuteMarketOrder`. Logs events (INIT, SIGNAL, TRADE) to journal for debugging.
- **Logic Details:**
1. **Initialization (OnInit):** - Set the trading timeframe to M5 explicitly. e.g. `ENUM_TIMEFRAMES tf = PERIOD_M5;`
- Call `InitIndicators(_Symbol, tf)` to create indicator handles for ATR, ADX, EMA(50), RSI. Check for success.
- Initialize global last bar time for M5 (`g_lastBar_M5`) to current bar time (so EA waits for next bar).
- Set up trade environment: ensure MagicNumber is set (via `EnsureTradeInit()` in orders) and any needed global flags (e.g., initialize `g_noTradeUntil = 0`).
- Log "OK - M5 Timeframe" on successful init.
2. **OnTick Handling:** - Retrieve current time `now = TimeCurrent()`. Update daily/weekly drawdown tracking via `TM_DD_Update(now)`.
- Manage open trades: call `TM_ManageOpenPositions()` each tick (this will handle trailing stops if enabled and also detect closed trades to update loss streak as per Task F).
- **New Bar Check:** Use `if(!NewBarGuard(PERIOD_M5, g_lastBar_M5)) return;` to only proceed on a new M5 candle. If it's not a new bar, do nothing further (just continue managing open positions).
- **Pre-Trade Gate:** Call `if(!PreTradeGate(now))` to enforce risk and session filters: - Inside PreTradeGate, call `TM_DD_GateOK()` (daily DD limit) and `TM_Week_GateOK()` (weekly DD) – if either fails, gate is false. - Then call `CanTradeNow(_Symbol, now)` – which checks Spread <= Max_Spread, ATR >= Min_ATR, allowed day & session time, margin level >= 200%, and not in no-trade pause (g_noTradeUntil). - If PreTradeGate returns false, log a “Blocked: ...” message with reasons (e.g., drawdown %, spread, ATR, session off, pause, etc.) and exit OnTick. No new trade attempted.
- **Build Signal:** If gate passed, analyze the new M5 bar for a trade signal: - Read indicator values: `double adx = ADX(1); double rsi = RSI(1); double ema = EMA(1);` for the last closed bar (index 1). - Determine **trend mode**: `bool trendMode = (adx >= ADX_Trend_Threshold);` and **bias direction**: if trendMode true, then if last close >= ema (and EMA slope upward) set bias=DIR_LONG, if last close <= ema (and slope downward) set bias=DIR_SHORT. If trendMode false (low ADX), set bias = DIR_NONE (range mode). *Edge case:* If adx is borderline or ema flat, treat as range (bias none) to be safe.
- Prepare a `Signal sig` struct and default `sig.valid=false`.
- If `trendMode == true`:
- If `bias == DIR_LONG`: check **Long entry conditions** – oversold in uptrend: if `rsi <= RSI_Buy_Threshold` (e.g. <=30) *and* there was at least one recent bearish bar. You can check the previous bar’s close vs its open or the previous bar’s close < close of 2 bars ago to ensure a pullback occurred.
- Also require a bullish reversal pattern: e.g. if the last closed bar is bullish and closed above the high of the previous bar (engulfing or breakout of pullback high).
- If all long conditions met: set `sig.valid=true; sig.dir=DIR_LONG; sig.reason=REASON_SCALP;` and compute stop-loss and take-profit:
- SL = low of the last 1-2 bars (the lowest point of the pullback) minus a small buffer (e.g. 5 points). Ensure SL

distance >= MIN_SL_POINTS.
- TP = entry + 1.5 * (entry - SL) (for ~1.5R target). Ensure TP distance >= MIN_TP_POINTS.
- Set `sig.entry = 0.0` (market entry at current price; the ExecuteMarketOrder function will use current Bid/Ask). Set `sig.sl` and `sig.tp` to the prices calculated.
- If `bias == DIR_SHORT` (downtrend): do analogous checks for Short entry – overbought in downtrend: `rsi >= RSI_Sell_Threshold` (e.g. >=70), plus at least one recent bullish bar indicating a pullback up, and a bearish reversal pattern on last bar (e.g. last bar closed below the low of previous bar). If conditions met, prepare `sig` similarly with DIR_SHORT, stop above recent swing high, TP 1.5R below entry.
- If `trendMode == false` (range mode, ADX low):
- Consider both long and short possibilities depending on price extremity:
- Check **Long setup**: if `rsi <= RSI_Buy_Threshold` (<=30, oversold) and price action shows a potential bottom (e.g. last bar bullish close above prior bar high). This suggests price reversed from a lower extreme. If so, form a long signal (sig.valid=true, dir=LONG, SL a few points below the recent low, TP at 1.5R).
- Check **Short setup**: if `rsi >= RSI_Sell_Threshold` (>=70, overbought) and price shows a top (last bar bearish close below prior bar low). If so, form a short signal (SL just above recent high, TP 1.5R down).
- *Edge case:* It's possible both long and short conditions trigger on the same bar if RSI crosses mid (unlikely with thresholds 30/70). Typically only one extreme will be true. If somehow both signals valid (extremely rare), you could prioritize one (e.g., whichever has higher RSI divergence or simply skip to avoid ambiguity). Generally, only one side will meet the threshold.
- After evaluating, you will have at most one `sig` with valid=true. If none of the conditions met, no trade this bar (exit OnTick without action beyond logging "No signal - Regime=TREND/Range ADX=X" if desired for debugging as v1 did).
- **Execute Trade:** If `sig.valid` came out true:
- Determine trade volume via `double lot = ComputeLot(ref_price, sig.sl)` where ref_price = current ask for buy or bid for sell (Calculate inside ExecuteMarketOrder as well or before). If lot <= 0 (e.g., too small volume), log "lot=0; skip" and abort signal.
- Call `ExecuteMarketOrder(sig, lot, ticket)` from orders.mqh. This will send a market buy or sell with given SL/TP.
- Check result: if sent returns true, log a success trade event ("SENT: ticket=... reason=SCALP lot=X"). If send fails, log error with `GetLastError()`.
- After sending, no further action in this tick (the trade_manager will handle any subsequent trailing management or closure tracking on later ticks).
- **Edge Cases & Error Handling:**
- Ensure not to generate multiple signals on the same bar: since we set Only_New_Bar true and we use one bar's data once, we naturally only attempt one trade per bar. Also, we enforce one position at a time, so if a trade is already open (`PositionsTotal()>0` with our magic), we should skip building a new signal. We can implement that check at the start of OnTick (after PreTradeGate). If an open position exists, skip new entries (our risk profile says one at a time).
- Indicator availability: sometimes RSI or ATR might return EMPTY_VALUE on the very first bar or if not enough history. Our functions handle that by returning false if not available. So if ATRSane or RSIConfirm fail due to no data, just do not trade. This is usually only during first few bars after EA start.
- Price pattern detection: be careful that we use last closed bar (index 1) for confirming pattern. We must not use index 0 (current forming bar) in logic since we trade on new bar open. Our use of NewBarGuard ensures index 1 refers to the bar that just closed.
- Spread widening right at execution: We rely on Max_Slippage_Points to avoid bad fills. If slippage is too high, ExecuteMarketOrder will fail and we log it; that trade is skipped.
- Logging: maintain helpful logs for blocked conditions, signals, and trades as in v1. E.g., if no signal found,

log “No signal - Regime=TREND ADX=35” or similar for debugging (not mandatory, but useful). If a signal found, maybe log “SIGNAL: Long scalp signal formed (RSI=28, Pattern=Engulfing)”. But main logs at least trade execution.
- **References:** Uses **Indicator values** from `indicators.mqh` (EMA, ADX, RSI, ATR). Conditions reference **threshold inputs** (ADX_Trend_Threshold, RSI_Buy/Sell_Threshold, Min_ATR_Points, etc.). Fulfills design requirements for entry conditions (trend + oscillator + pattern) and ensures risk management via PreTradeGate (drawdown & session checks) and one-trade-at-time enforcement (Risk Profile 5.4).

Task B – Revise Configurable Inputs (config_inputs.mqh)
Description: Modify the EA’s user input parameters to align with the new strategy and constraints. Remove obsolete v1 settings and introduce new ones for v2.
- **Inputs to Remove:** All parameters specific to v1’s breakout and liquidity-sweep logic are eliminated: - `Breakout_Lookback, Breakout_CloseBeyond_Points, Pending_Offset_Points, Breakout_Use_ADX_Filter, Breakout_Use_Volume_Filter, Vol_MA_Period, Vol_Min_Ratio` (no longer used in v2). - `Range_Lookback_Bars, Range_Min_Width_Points, Range_Max_Width_Points, Sweep_Buffer_Points, Sweep_Reentry_Confirm_Bars, Sweep_Use_RSI_Confirm` (no longer needed). - These should be fully removed or commented out to avoid confusion, since v2 logic doesn’t require them.
- **Inputs to Modify (Defaults):** Update defaults of retained inputs to match v2 design: - `EMA_Trend_Period`: change default from 200 to **50** (our primary trend EMA on M5). - Optionally, if we want a long-term EMA too, we could introduce `EMA_Long_Period=200`. If not, just use EMA_Trend_Period=50 and assume one EMA is enough. - `ADX_Trend_Threshold`: lower default from 25 to **20.0** for sensitivity (the threshold between trend vs range). - `RSI_Buy_Threshold`: set to **30** (was 35). `RSI_Sell_Threshold`: set to **70** (was 70, keep or adjust symmetrically). We choose 30/70 which are classic oversold/overbought levels to trigger entries. - Remove `RSI_Exit` (not used in v2). - `ATR_Period`: keep 14. - `Min_ATR_Points`: keep 100 (so we don’t trade when ATR14 < $1 range). - `Max_Spread_Points`: change default from 1500 to **50** (meaning 50 points = $0.50). This is the hard spread cap allowed. - Risk and Drawdown: - `Risk_Mode`: default to **1** (Percent risk mode by default now). - `Risk_Percent`: reduce default from 2.0 to **0.5** (% of equity per trade). This aligns with our target risk per trade. - `Max_Daily_Drawdown_Percent`: reduce from 5.0 to **3.0** (%). - **Add** new input `Max_Weekly_Drawdown_Percent = 7.0;` (to enforce weekly loss limit). - **Add** new input `Consecutive_Loss_Limit = 10;` (number of losses before pause). - **Add** new input `Loss_Pause_Duration_Min = 120;` (pause 120 minutes = 2 hours after hitting loss streak). - Session and Time: - `Use_Session_Filter`: keep true (we will use Session_Start/End). - `Session_Start_Hour`: keep 7. - `Session_End_Hour`: keep 18 (so trading 07:00–18:00 GMT). - `Friday_CloseTime`: keep "21:00" (EA will stop trading after 21:00 GMT Friday and close positions). - Trading days Mon–Fri: keep all true (we allow trading each weekday, with Friday being partial day via Friday_CloseTime). - Trade Management: - `Trail_Mode`: default **0** (no trailing by default, Option A). - `Move_BE_After_Points`: can remain 150 (points) but since trailing is off, doesn’t matter. - `Use_Partials`: default **false** (no partial closes by default). - We keep these management inputs available but off, so that advanced users can enable trailing or partial if desired in the future (which would approximate Option B logic). - Profit Target: - `TP_Mode`: remain 1 (RR mode). - `RR_Target`: change default from 2.0 to **1.5** (we aim for 1.5R TP). - `TP_Fixed_Points`: keep 200 (not used by default since TP_Mode=1, but user can switch to fixed pips if wanted). - Housekeeping: - `Order_Comment`: update string from "XAUUSD-GOD-M15" to **"XAUUSD-GOD-M5"** or "XAUUSD-GOD-v2". This will tag orders clearly for M5 version. - `Magic_Offset`: keep as 1 (user can change if running multiple instances). - `Max_Slippage_Points`: keep 50 (points). Already set in v1. -

`Order_Expiration_Min`: keep 30 (though we aren't using pending orders by default). - Remove any input not used (e.g., v1's Only_New_Bar and Order_Type if they were, but we actually still use Only_New_Bar). - Actually, keep `Only_New_Bar = true` (we still rely on it for NewBarGuard). - `Order_Type` was not actively used (could have been intended to toggle between market/pending?). We can remove `Order_Type` to avoid confusion, since v2 always uses market orders. If it isn't used anywhere in code, it's safe to remove. - **Edge Cases & Remarks:** - Ensure input changes reflect in corresponding logic: e.g., new `Max_Weekly_Drawdown_Percent` input will be used in trade_manager's weekly gate, `Consecutive_Loss_Limit` and `Loss_Pause_Duration_Min` used in loss streak logic. - Removing the breakout/sweep inputs will not break anything as long as we removed their usage in code. We should double-check no remaining code references them. - The input defaults chosen aim for safety and moderate trading frequency. Users can still adjust if needed, but these align with our research benchmarks (e.g., 0.5% risk, 1.5R target, small daily DD). - Group the new inputs appropriately for clarity: - We may create a new group "Risk Limits" for daily/weekly drawdown and streak pause settings. For example:

```
input group "Risk Limits"
input double Max_Daily_Drawdown_Percent = 3.0;
input double Max_Weekly_Drawdown_Percent = 7.0;
input int Consecutive_Loss_Limit = 10;
input int Loss_Pause_Duration_Min = 120;
```

This keeps them organized in UI. - By streamlining inputs, the EA is easier to configure for v2. We ensure every input is either utilized in logic or clearly optional (like trailing/partials, which are off by default).

Task C – Enhance PreTradeGate and CanTradeNow (spread_vol_filter.mqh)
Description: Expand the gating functions to include session timing, margin level, and loss-streak pause checks, in addition to existing spread/ATR/day filters. These functions decide whether the EA can attempt a new trade on the current tick.
- **Inputs:** They use various global inputs: `Max_Spread_Points`, `Min_ATR_Points`, `Use_Session_Filter`, `Session_Start_Hour`, `Session_End_Hour`, `Friday_CloseTime`, and our new risk limit states (like g_noTradeUntil for pause, and margin threshold). Also relies on current server time passed in.
- **Outputs:** Returns boolean (true = safe to trade now, false = do not trade). If false, PreTradeGate will compile a reason string for logging.
- **Logic in `CanTradeNow(symbol, time)`:**
1. **Spread Check:** Compute current spread in points (difference between Ask and Bid divided by Point). If `Max_Spread_Points > 0` and actual spread > Max_Spread_Points, return false (spread too high). If spread info not available (SymbolInfoTick fails), be conservative and return false (can log an error).
2. **Volatility Check:** If `Min_ATR_Points > 0`, get ATR(1) via `ATR(1)` function. If ATR is `EMPTY_VALUE` (indicator not ready), treat as not okay to trade (to avoid trading blindly). If ATR < Min_ATR_Points, return false (market too quiet).
3. **Day/Session Check:** - Use `DayAllowed(time)` to ensure it's an allowed weekday. If that returns false (like weekend or user disabled the day), return false.
- If `Use_Session_Filter` is true: enforce intraday session hours. Convert `time` (server time) to hour:minute (use MqlDateTime). If `time.hour < Session_Start_Hour` or `time.hour >= Session_End_Hour`, return false (outside trading window for that day).
- Additionally, if it's Friday (`day_of_week == 5`) and `time.hour >=` Friday_CloseTime hour (e.g., 21)

and minute >= close minute, return false (don't allow new trades after Friday close time). Essentially, after 21:00 GMT Friday, no new trades.
4. **Margin Level Check:** Retrieve `double marginLevel = AccountInfoDouble(ACCOUNT_MARGIN_LEVEL)`. If `marginLevel !=0` (broker provides it) and `marginLevel < 200` (or `< Minimum_Margin_Level_Percent` input if we added one, default 200%), then return false. This means account margin is too low (e.g., heavy drawdown or other positions consuming margin), so we avoid opening more trades.
5. **Consecutive Loss Pause Check:** Check the global pause flag/time (e.g., `g_noTradeUntil`). If we have a pause active (`g_noTradeUntil > TimeCurrent()`), then return false (currently in cooling-off period after too many losses).
- If all checks pass, return true (trading allowed this tick).
- **Logic in** `PreTradeGate(time)`**:** (This function is in XAUUSD-GOD.mq5, but referencing here how to update it together with filters): - PreTradeGate will call `TM_DD_GateOK()` (daily drawdown limit) and our new `TM_Week_GateOK()`. If either returns false (drawdown exceeded), it immediately returns false.
- Then it calls `CanTradeNow(_Symbol, time)`. If that returns false (for any of the above reasons), PreTradeGate returns false.
- If all checks are okay, PreTradeGate returns true.
- PreTradeGate also builds a detailed log reason string when returning false: We will append messages for each cause: - If `!TM_DD_GateOK()`: append `"DD=" + current_drawdown + "% "`. (We get current drawdown from `DD_CurrentPercent()` in trade_manager). Similarly, if `!TM_Week_GateOK()`, append `"WD=" + week_drawdown + "% "` (to indicate weekly drawdown triggered).
- Always append current Spread and ATR values for context: e.g., `"Spread=" + spread_points + " ATR=" + atr_value`. This is done regardless of their thresholds for debugging (like v1 did). If ATR or spread not available, skip or indicate.
- If session filter blocked (we can know by checking hour vs range): append `"SessionOff"`. If day not allowed: append `"DayOff"`.
- If margin too low: append `"Margin=" + (int)marginLevel + "%"`.
- If in loss pause: append `"Paused"`.
- This way, when a trade is blocked, the log might read: *"Blocked: DD=3.5% Spread=25 ATR=0.50 Paused"* (for example, indicating daily DD hit and pause active), or *"Blocked: Spread=60 ATR=1.20 SessionOff"* (spread too high and outside session).
- Ensure spacing and formatting is clear. We gather these by checking conditions similarly to how CanTradeNow did or by reusing flags. Simpler: we can replicate checks in logging: - e.g., if SpreadOK returned false or simply if spread_points > Max_Spread then we know it was cause. - if ATR < Min_ATR then maybe add "ATRLow". - But we decided to simply always output spread & ATR values. That's fine. For session, margin, pause, we should only append if they triggered: - For session: if Use_Session_Filter true and (time outside hours or Friday after close), we append "SessionOff". - For margin: if marginLevel < 200% append "MarginLow". - For pause: if g_noTradeUntil now in future, append "Paused". - For day disallowed or weekend: perhaps "MarketClosed" or "DayOff" if we detect dt.day_of_week in {0,6} or Trading_Day_x false. But since typically EA won't even run ticks on weekend, not critical. If user disabled Wednesday (Trading_Day_Wed=false), we'd block trades that day, might as well log "DayOff" in blocked reason. We can detect if !DayAllowed and add "DayOff". - This enriched logging will help the user understand why the EA isn't trading at certain times.
- **Edge Cases:**
- If spread or ATR info isn't available (e.g., right at EA start before indicators ready), `CanTradeNow` will return false (conservative approach) but PreTradeGate logging should handle not having values (we won't append ATR if ATR(1) returned EMPTY, etc.). That might happen first tick or two after initialization. It's fine;

EA will wait.
- Margin check: We assume if AccountInfoDouble returns 0 or huge value when no positions, marginLevel might be not meaningful. We only act if marginLevel is positive but below threshold. If no positions and marginLevel is e.g. 10000%, it won't block. If user is close to margin call with other trades, it will block new trades appropriately.
- Loss pause: `g_noTradeUntil` is set when the loss streak triggers (Task F). It's a static datetime. Ensure it's initially 0 and gets set properly. The check in CanTradeNow should handle if TimeCurrent() is slightly beyond the threshold – once current time >= g_noTradeUntil, the pause is over and trades resume.
- Session times on different days: We compare raw hour, but for final hour of session e.g. 18:00, since we use `>= Session_End_Hour` as out-of-session, it means at exactly 18:00 it stops (no new trades on or after 18:00). That effectively means last trade can start at 17:55 and that's fine. That is our intent (stop at 18). Similarly for Friday 21:00 – at 21:00 we stop. If we wanted allow up to 20:59 inclusive, this logic does that.
- Daylight savings: We assume server time GMT consistent. If not, user might adjust session hours accordingly. It's beyond code scope. We just provide hour-based filter.
- **Testing scenarios:** - Monday morning before session start: expected block with "SessionOff". - Middle of night (if EA left running 00:00): block with "SessionOff" (if still on same day). - Friday 21:05: block with "SessionOff" (past Friday close). - If user manually disables a day in inputs: if that day, block with "DayOff". - If user running other trades causing margin <200: block with "MarginLow". - After 10 losses: we will see "Paused" for 2 hours in logs. - On normal conditions, PreTradeGate returns true and we get "No signal..." or a trade.

Task D – Implement Weekly Drawdown Tracking (trade_manager.mqh)
Description: Extend the drawdown tracking system to cover weekly drawdown in addition to daily. This ensures the EA stops trading for the week if losses exceed the weekly limit.
- **Inputs:** Uses `Max_Weekly_Drawdown_Percent` from inputs (e.g., 7%). Also uses account equity via `AccountInfoDouble(ACCOUNT_EQUITY)`. Relies on time via `TimeCurrent()` to detect new week.
- **Outputs:** Maintains static variables `g_week_start_equity` and `g_week_peak_equity` and provides a function to check weekly DD (TM_Week_GateOK). This will be used in PreTradeGate to potentially block trades.
- **Changes to TM_DD_Update(datetime tv):**
- After existing daily reset logic, add logic to reset weekly stats at start of week:

```
MqlDateTime dt; TimeToStruct(tv, dt);
if(dt.day_of_week == 1 || g_dd_day < 0) { // Monday or first run
    // Determine if it's a new week
    if(g_dd_week < 0 || dt.day_of_week < g_last_day_of_week) {
        // If last known day was Sunday (0) or we've rolled over weekend
        g_week_start_equity = AccountInfoDouble(ACCOUNT_EQUITY);
        g_week_peak_equity = g_week_start_equity;
    }
}
g_last_day_of_week = dt.day_of_week;
```

Alternatively, simpler: if NewServerDay triggered *and* it's Monday, then reset weekly. Because daily resets every day, we can embed weekly reset in Monday's daily reset. For example:

```
if(NewServerDay(tv) || g_dd_day < 0) {
    DD_Reset(tv); // resets daily
    if(dt.day_of_week == 1) {
        // It's Monday
        g_week_start_equity = AccountInfoDouble(ACCOUNT_EQUITY);
        g_week_peak_equity = g_week_start_equity;
    }
}
// Then update peaks:
double eq = AccountInfoDouble(ACCOUNT_EQUITY);
if(eq > g_week_peak_equity) g_week_peak_equity = eq;
```

Also define a static or global to store something like last week if needed. Actually, above logic covers it: each Monday we reset. If EA starts mid-week, g_week_start_equity gets set on that Monday presumably (or if starting Tue, the condition dt.day_of_week==1 fails so it won't set, meaning it erroneously carries initial equity as from perhaps previous Monday? Actually on start, g_week_start_equity not set; maybe we should initialize it on EA start as well: - If g_week_start_equity is 0 initially and dt.day_of_week != 1, we might set week_start to current equity on EA start to avoid undefined. We can do: if(g_week_start_equity == 0) set it now (meaning EA started mid-week, so use current equity as baseline for rest of week).)
- After updating g_week_peak_equity each tick (like they do for daily peak: `if(eq > g_dd_peak_equity) g_dd_peak_equity = eq;`), do similarly for weekly: `if(eq > g_week_peak_equity) g_week_peak_equity = eq;`.
- **New function TM_Week_GateOK():**
- Define `double week_dd = 0.0;`
- If `Max_Weekly_Drawdown_Percent <= 0` return true (disabled). Otherwise compute current weekly drawdown percent:

```
if(g_week_peak_equity > 0) {
    double eq = AccountInfoDouble(ACCOUNT_EQUITY);
    week_dd = (g_week_peak_equity - eq) / g_week_peak_equity * 100.0;
    if(week_dd < 0) week_dd = 0;
}
return (week_dd < Max_Weekly_Drawdown_Percent);
```

- This mirrors TM_DD_GateOK which did similar for daily (peak vs current equity comparison) [95] . - We might also provide a function to get current weekly DD percent for logging (like v1 had DD_CurrentPercent for daily). We can add `double WeekDD_CurrentPercent()` (similar to existing `DD_CurrentPercent()`) to use in PreTradeGate logging to append "WD=X%".
- **Integration:**
- Ensure TM_Week_GateOK is called in PreTradeGate: e.g.,

```
if(!TM_DD_GateOK() || !TM_Week_GateOK()) return false;
```

- Also update PreTradeGate logging to include weekly drawdown if triggered:

```
if(!TM_DD_GateOK()) reason += "DD=" + DoubleToString(DD_CurrentPercent(),2) + "%
";
if(!TM_Week_GateOK()) reason += "WD=" + DoubleToString(WeekDD_CurrentPercent(),
2) + "% ";
```

This way, if weekly loss limit hit, it logs WD=7.0% for instance.
- **Edge Cases:**
- EA start mid-week: The weekly baseline might not be properly set by above logic (since we only set on Monday). To handle this: when EA first runs (g_dd_day < 0 triggers daily reset), we do check if it's Monday or not. If not Monday, we could set g_week_start_equity = current equity as an initial baseline (so that if losses happen from now till Sunday, it measures relative to start). Otherwise, if we don't do that, `g_week_peak_equity` remains 0 and our week DD calculation might be off (because it won't update properly). - Could do:

```
if(g_week_peak_equity <= 0) {
    g_week_start_equity = eq;
    g_week_peak_equity = eq;
}
```

on EA start (perhaps put under g_dd_day < 0 branch regardless of day). - End of week: Our reset triggers Monday. If the EA isn't running on Monday at exactly midnight but gets first tick at Monday 01:00 for example, daily reset code will handle Monday. If market is closed Monday (rare) and opens Tuesday, our logic won't reset weekly because day_of_week is 2. That means it would still think it's same week. This is an edge case for e.g. Christmas week. Possibly acceptable, as weekly limit might not be needed in holiday scenario. But for correctness, we might consider if dt.day_of_year difference is >3 from last day of trading, treat as new week anyway. Given complexity, we accept minor anomaly. - The computed weekly drawdown uses highest equity of week. If a user deposits or withdraws mid-week, the peak equity calculation would incorporate that jump. Our logic might interpret deposit as new peak and thus not hindering trading (which is fine). We just note that any deposit raises peak equity, making it harder to hit weekly DD% (which is acceptable). - If weekly limit is hit on a Friday, the EA will stop anyway and then Monday resets everything, which is fine.
- **Testing:** Simulate weekly progression in backtest or by adjusting the date and equity manually. Ensure that when equity falls 7% from Monday’s peak, TM_Week_GateOK returns false and PreTradeGate blocks trades (with log "WD=7.0%"). Ensure Monday’s equity is taken as baseline and that by next Monday it resets.

Task E – Enforce One-Trade-at-Time & Loss Streak Pause (trade_manager.mqh)
Description: Implement logic to ensure the EA has at most one open position at any time and to handle the consecutive loss count and trading pause mechanism.
- **One Position at a Time:** - Add a check in the main OnTick (Task A) after PreTradeGate passes but before building a new signal:

```
// in OnTick after PreTradeGate:
int myPositions = 0;
for(int i=PositionsTotal()-1; i>=0; --i) {
    if(PositionSelectByIndex(i)) {
        if(PositionGetInteger(POSITION_MAGIC) == MAGIC_BASE + Magic_Offset &&
           PositionGetString(POSITION_SYMBOL) == _Symbol) {
            myPositions++;
        }
    }
}
if(myPositions > 0) {
    // Already an open position for this EA, skip new signal
    return;
}
```

(We can simplify by storing open trades count at global each tick, but loop is fine since typically 0 or 1 positions.) - Alternatively, we can rely on `CanTradeNow` margin or logic because if a position is open, our margin is used but margin level could still be high if risk small, so margin check won't necessarily block. So we explicitly block by checking open positions as above. - This ensures no new trade is created if one is already open. This is in line with "Only ONE position open at a time" in risk profile.
- Edge: If user somehow manually closes a trade or multiple positions exist from manual intervention (shouldn't for same magic), the loop covers it. If using different Magic_Offset on multiple charts, each EA sees only its own positions due to magic match. - **Consecutive Loss Tracking:** - Introduce static counters in trade_manager (top of file):

```
static int g_loss_streak = 0;
static bool g_position_was_open = false;
```

These persist across ticks. g_loss_streak counts consecutive losses, g_position_was_open is a flag indicating whether we had a position open on the previous tick.
- In `TM_ManageOpenPositions()` (called each tick in OnTick): - At the start or end, detect closing of a trade:

```
int count = 0;
// count current open positions for this EA (similar loop as above or use
PositionsTotal and Magic filter)
for(int i=PositionsTotal()-1; i>=0; --i) {
    if(PositionSelectByIndex(i)) {
       if(PositionGetInteger(POSITION_MAGIC) == MAGIC_BASE + Magic_Offset &&
          PositionGetString(POSITION_SYMBOL) == _Symbol) {
           count++;
       }
    }
}
// if previously there was an open pos and now count is 0, a position closed
if(g_position_was_open && count == 0) {
    // Determine outcome of last closed trade
    double profitSum = 0;
    // Use HistoryDeal to find last closed deal for our magic+symbol:
    datetime since = TimeCurrent() - 86400; // look back last 24h
    HistorySelect(since, TimeCurrent());
    long totalDeals = HistoryDealsTotal();
    for(long i = totalDeals-1; i >= 0; --i) {
        ulong dealTicket = HistoryDealGetTicket(i);
        ulong dealMagic = HistoryDealGetInteger(dealTicket, DEAL_MAGIC);
        string dealSymbol = HistoryDealGetString(dealTicket, DEAL_SYMBOL);
        long dealEntryType = HistoryDealGetInteger(dealTicket, DEAL_ENTRY);
        if(dealMagic == MAGIC_BASE + Magic_Offset && dealSymbol == _Symbol &&
dealEntryType == DEAL_ENTRY_OUT) {
            double dealProfit = HistoryDealGetDouble(dealTicket, DEAL_PROFIT) +
HistoryDealGetDouble(dealTicket, DEAL_COMMISSION) +
HistoryDealGetDouble(dealTicket, DEAL_SWAP);
            profitSum += dealProfit;
            // break after summing (though partial closings would generate
multiple deals, profitSum accumulates them)
            // In our case with no partial, one deal corresponds to entire trade
closure
            break;
        }
    }
    if(profitSum < -0.0001) {
        // loss
        g_loss_streak++;
    } else if(profitSum > 0.0001) {
        // win
        g_loss_streak = 0;
    } else {
        // roughly 0 (breakeven)
        g_loss_streak = 0;
    }
    // Check loss streak vs limit
    if(g_loss_streak >= Consecutive_Loss_Limit && Consecutive_Loss_Limit > 0) {
        g_loss_streak = 0; // reset streak count after triggering pause
        g_noTradeUntil = TimeCurrent() + Loss_Pause_Duration_Min*60;
        // Log pause event
        LogEvent("TM", "Max loss streak reached, pausing new trades for " +
IntegerToString(Loss_Pause_Duration_Min) + " min");
    }
}
// update flag for next tick
g_position_was_open = (count > 0);
```

- This code scans trading history for the last deal with our magic and symbol where DEAL_ENTRY == DEAL_ENTRY_OUT (meaning a closing trade action). It sums profit from that deal (including commission, swap to get net). Since we have no partial closures by default, one closing deal corresponds to the whole trade result. If partials were used, the loop might catch the final partial or multiple deals – here we break after one because we assume either one final deal (no partial) or we break to avoid counting separate

partial deals as separate trades (we treat the entire sequence as one trade outcome, which might undercount losses if partial was lost later, but optional scenario). In Option A, it's straightforward: one close, one profitSum.
- If profitSum is clearly positive -> it was a win, reset loss streak. If clearly negative -> loss, increment streak. If ~0 -> treat as breakeven (reset streak to 0, not counting as loss). We use a small epsilon 1e-4 to avoid floating rounding issues.
- If loss streak reaches the threshold (10), we: - Reset streak to 0 (so the count starts fresh during pause or after). - Set the global `g_noTradeUntil` to current time + pause duration (in seconds). We use the Loss_Pause_Duration_Min input for how many minutes to pause.
- We log an event indicating the pause activated (e.g., "Max loss streak reached, pausing new trades for 120 min").
- We rely on `g_noTradeUntil` (declared as external static in main or trade_manager) being checked in CanTradeNow (Task C) to actually block trades during this pause window.
- Also ensure `g_noTradeUntil` is declared accessible: - We decided in Task C to declare `static datetime g_noTradeUntil = 0;` in the global scope of the main .mq5 file (after includes). So here in trade_manager we just reference it (since it's in the same program, linking is fine). - `Consecutive_Loss_Limit` and `Loss_Pause_Duration_Min` come from config inputs, so ensure trade_manager has access to those (it includes config_inputs so yes). - No output directly from ManageOpenPositions beyond modifying global g_noTradeUntil and logging. - **Edge Cases & Efficiency:**
- We check history deals every time a trade closes. HistorySelect(24h) is fine as we trade rarely; at most 1-2 trades per day. The loop of deals runs backwards until it finds one matching our EA. This is efficient enough given few deals. - Partial close scenario (if user enabled partials in future): The code as written would break after the first closing deal it finds. In partial closes, there could be multiple deals (one for partial close, one for final close). Our code might then catch only the final close deal (since we break after one). Actually, since we break at the first found `DEAL_ENTRY_OUT`, and deals are iterated backwards (latest first), the *latest* close deal is likely the final portion closing. So profitSum will equal that final portion's profit, not the whole trade's profit. For partial logic, we might accumulate all deals belonging to one position. But for now (no partial by default), one deal = one trade. - If user manually closed partially, that could confuse results, but that's advanced usage beyond our main design. If partials/trailing are off, one full close covers it. - We reset streak after pause triggers to avoid immediate re-trigger after pause. That means effectively we treat hitting 10 losses and pausing as an intervention that breaks the streak. This is fine: after pause, if 10 more losses happen, it will pause again. - `g_position_was_open` ensures we only evaluate streak when a position actually closed and was previously open. If EA hasn't taken a trade yet or if it's continuously open (like in trailing for hours), it won't incorrectly adjust streak. - We consider any net positive as win, net negative as loss. A tiny profit from e.g. breakeven SL might count as win (resets streak). That's acceptable. We just don't want to count it as continuing loss. - Logging: The pause log uses `LogEvent("TM", "...")` to annotate the risk management. This helps user see in logs when a pause started. We also log in PreTradeGate if pause is active (with "Paused"). Combined, user knows how long and when. - We must include `Trade.mqh` or relevant enumerations for `DEAL_ENTRY_OUT` (which is an MT5 enum for trade deals). If not included, we define `#define DEAL_ENTRY_OUT 1` but better include `<Trade\Trade.mqh>` at top (which is likely included already via orders or main). In our codebase, orders includes `<Trade/Trade.mqh>`, so by the time trade_manager is compiled, it should know those constants. - **Testing:** Simulate a sequence of losing trades in backtest or via parameter modifications: - We can artificially increment g_loss_streak in code for test or trigger trades with known outcomes. - After 10 losses, ensure `g_noTradeUntil` sets ~2h and PreTradeGate blocks trades during that window. - After 2h passes in test (fast-forward time), ensure EA resumes trading (loss streak was reset). - Also test that a single win resets the streak to 0 as expected. - Confirm that if EA is restarted (which resets static g_loss_streak to 0

since not saved), it doesn't erroneously remain paused (g_noTradeUntil would also be reset to 0 on reinit as we declared it static in main globally - actually static in global scope persists only during runtime; on EA reinit it resets too). That means if EA is reloaded during a pause window, it will lose memory of pause and possibly trade again. This is a known limitation (since we don't persist state through EA restart). It's acceptable; we assume user not doing that to circumvent logic. We could mitigate by writing pause end time to a file or global variable (GlobalVariableSet) so it persists on restart. That's an enhancement beyond current scope but possible. For now we note that reinitializing EA clears the pause.

Task F – Remove Unused Modules (liquidity_sweep.mqh, breakout.mqh, range_detector.mqh)
Description: Delete or exclude the files containing v1's obsolete strategy logic to avoid confusion and ensure they are not referenced. Clean up any remaining references in code and project.
- **inc/liquidity_sweep.mqh:** This file implemented the old range liquidity sweep strategy. In v2 it's fully replaced by our unified scalp logic.
- **Actions:** Remove `#include <inc/liquidity_sweep.mqh>` from the main EA file (XAUUSD-GOD.mq5) [96]. - Delete or comment out the file from the project so it's not compiled. - Ensure no references remain: v1 had calls to `ScanAndSignal_LiquiditySweep()` [97] and used constants like `REASON_LIQ_SWEEP`. We have removed those calls in Task A and replaced reason codes. So there should be no unresolved references once include is gone. - No inputs tied specifically to this (except Sweep_Use_RSI_Confirm etc., which we removed in Task B). - Check that the `REASON_LIQ_SWEEP` constant from constants.mqh is removed or not used, as we replaced with REASON_SCALP. - **inc/breakout.mqh:** This contained the trend breakout signal logic. Also not used in v2. - **Actions:** Remove `#include <inc/breakout.mqh>` from main file [96]. - Remove calls to `ScanAndSignal_Breakout()` in BuildSignal (Task A did that) [98]. - Remove or inactivate any constants in constants.mqh used only by breakout (like `REASON_TREND_BO`). We replaced references with our own or removed them. - Delete the file or exclude from build. - Check config inputs for Breakout_Use_Volume_Filter etc. removed in Task B. Ensure no stray references (like in filters.mqh VolumeConfirm was partly tied to volume filter input which we set not used but left function; that's fine as long as nothing calls VolumeConfirm now). - **inc/range_detector.mqh:** This was used by liquidity_sweep to find consolidation range. - **Actions:** Remove `#include <inc/range_detector.mqh>` from main file [99]. - No calls to `BuildRangeBox` remain (since liquidity sweep gone). - Delete or exclude file. - **General Clean-up:** After removing, compile the project to ensure no undefined references: - If any references to `Regime r` or `REGIME_RANGE/REGIME_TREND` remain but we removed includes that define them (rangeDetector didn't define regime, regime was separate and we kept Regime in types). Actually, we kept Regime enum in types and DetectRegime function in regime.mqh. We might still use DetectRegime for ADX threshold logic if needed, but if not, we can also remove regime.mqh usage entirely and just use ADX directly. - If we chose not to use DetectRegime anymore, we can also remove `#include <inc/regime.mqh>` and its usage. Alternatively, we can keep it if we want (though our code in Task A didn't call DetectRegime, we manually did ADX check). - It's fine to remove regime.mqh to simplify: we already integrated ADX logic manually. So optionally: - Remove `#include <inc/regime.mqh>` from main. - Remove any references to `Regime` enum. We used Regime in v1 mostly for logging and branching. In our new code, we can derive regime from trendMode bool when logging (e.g., log "Regime=TREND" if trendMode else "Regime=RANGE"). We don't need the Regime type explicitly. - So yes, we can drop regime.mqh as well for clarity. - Check that after removal, our logging or conditions don't rely on those. In PreTradeGate logging, v1 logged regime string via `Regime r = DetectRegime()` and message `"Regime="+(r==REGIME_RANGE?"RANGE":"TREND")` [100]. If we removed regime, we can replicate that easily: we have adx and threshold, if adx>=threshold log "TREND" else "RANGE". Not critical to log regime in final, but we can as info. Up to us. Could include in "No signal" log maybe. But not needed for functionality. - Remove any leftover variables or code related to these old modules (e.g., global `g_lastBar_M15` is

replaced by g_lastBar_M5, we can rename it). - Removing these modules reduces code bloat and ensures no accidental use of old logic.
- **Testing:** After removal, run a full compile. Ensure no references to `ScanAndSignal_LiquiditySweep`, `...Breakout`, `BuildRangeBox`, `REASON_LIQ_SWEEP`, etc. remain. The compile should succeed with no unresolved symbols. Then test EA to confirm it functions solely with new logic.
- **Edge Cases:** Removing these modules should have no runtime side effects, as we've replaced their functionality fully. The only risk is if something else indirectly relied on them (unlikely). For example, VolumeConfirm in filters was used only if Breakout_Use_Volume true which we removed. We left VolumeConfirm function but it's never called now, which is fine. We can also remove VolumeConfirm if we want absolute clean up (since no one calls it after breakout removal). - Let's remove VolumeConfirm too for cleanliness: it was called in breakout logic, not used now. Delete or comment out `bool VolumeConfirm()` in filters.mqh if desired. - That simplifies maintenance.

By completing the tasks above (A through F), we will have effectively transformed the EA from the v1 structure to v2 structure: - The main control flow now implements our new scalping strategy conditions. - All parameters and filters reflect the new approach (session limits, risk controls, etc.). - Safety mechanisms (loss streak pause, weekly stop) are in place as specified. - Legacy code is removed, preventing confusion or conflict. - The output (compiled EA) should now naturally produce ~100-300 trades/year on XAUUSD M5, with each trade managed as per Option A, and abiding by all risk constraints from section 5.

Each task above can be taken up by a coding assistant or developer in sequence (or parallel where independent), and together they cover the full implementation of version 2.

---

1 39 69 GitHub
https://github.com/techly39/XAUUSD-God-MT5-EA/blob/8a0fd0267634b9b815c1a698a62eecb3c40cab91/inc/regime.mqh

2 3 4 5 31 GitHub
https://github.com/techly39/XAUUSD-God-MT5-EA/blob/8a0fd0267634b9b815c1a698a62eecb3c40cab91/inc/liquidity_sweep.mqh

6 7 8 9 10 11 GitHub
https://github.com/techly39/XAUUSD-God-MT5-EA/blob/8a0fd0267634b9b815c1a698a62eecb3c40cab91/inc/breakout.mqh

12 13 14 15 43 63 79 80 GitHub
https://github.com/techly39/XAUUSD-God-MT5-EA/blob/8a0fd0267634b9b815c1a698a62eecb3c40cab91/inc/risk.mqh

16 18 19 20 44 91 92 93 94 95 GitHub
https://github.com/techly39/XAUUSD-God-MT5-EA/blob/8a0fd0267634b9b815c1a698a62eecb3c40cab91/inc/trade_manager.mqh

17 40 59 60 61 62 64 GitHub
https://github.com/techly39/XAUUSD-God-MT5-EA/blob/8a0fd0267634b9b815c1a698a62eecb3c40cab91/inc/config_inputs.mqh

21 26 27 28 30 45 46 47 48 49 50 51 52 96 97 98 99 100 GitHub
https://github.com/techly39/XAUUSD-God-MT5-EA/blob/8a0fd0267634b9b815c1a698a62eecb3c40cab91/XAUUSD-GOD.mq5

22 23 37 78 GitHub
https://github.com/techly39/XAUUSD-God-MT5-EA/blob/8a0fd0267634b9b815c1a698a62eecb3c40cab91/inc/spread_vol_filter.mqh

24 25 41 42 70 71 72 73 74 75 76 77 GitHub
https://github.com/techly39/XAUUSD-God-MT5-EA/blob/8a0fd0267634b9b815c1a698a62eecb3c40cab91/inc/filters.mqh

29 38 55 81 82 83 84 85 86 87 88 89 90 GitHub
https://github.com/techly39/XAUUSD-God-MT5-EA/blob/8a0fd0267634b9b815c1a698a62eecb3c40cab91/inc/orders.mqh

32 33 Trade Gold for +10 pips Daily | Scalping strategy | Forex Factory
https://www.forexfactory.com/thread/1274831-trade-gold-for-10-pips-daily-scalping

34 35 Spread & Slippage in Gold: What to Expect and How to Trade It - FX Signals – Gold Signals
https://www.fxpremiere.com/spread-slippage-in-gold-what-to-expect-and-how-to-trade-it/

36 Ultra Scalper EA for XAUUSD - My Trading - 7 August 2025 - Traders' Blogs
https://www.mql5.com/en/blogs/post/763788

53 68 GitHub
https://github.com/techly39/XAUUSD-God-MT5-EA/blob/8a0fd0267634b9b815c1a698a62eecb3c40cab91/inc/indicators.mqh

54 56 57 58 GitHub
https://github.com/techly39/XAUUSD-God-MT5-EA/blob/8a0fd0267634b9b815c1a698a62eecb3c40cab91/inc/constants.mqh

65 66 67 GitHub
https://github.com/techly39/XAUUSD-God-MT5-EA/blob/8a0fd0267634b9b815c1a698a62eecb3c40cab91/inc/utils.mqh