

QA-Testplan (Programmierer) — DowHowSignalService 2.010

Version: v2 (detaillierter Testkatalog) • Datum: 2026-01-10

Ziel: Den aktuellen Quellcode systematisch durchtesten (Ablauf, Objektmodell, Eventhandling, DB-Persistenz, Discord/Webhook, UI-Panels). Dieses Dokument ist für Entwickler gedacht: zu jedem Test steht was geprüft wird, wo im Code die Logik sitzt, wie man testet und woran man Pass/Fail erkennt.

Testumgebung (ausfüllen)	
MT5 Build / Broker	_____
EA-Datei	Trade Assistent V2.013.mq5/.ex5
Symbol / TF	_____
Inputs (relevante)	InpDebug=____ InpRequireKnownSymbol=____ Webhooks gesetzt? ____
Log-Datei	MQL5/Files/DowHowSignalService.log (Logger: CLogger::SetMethod(LOGGING_METHOD
DB-Datei	MQL5/Files/DowHowState_<SYMBOL>_<TF>.sqlite (CDBService::DefaultFile)

Wie man Logs/DB korrekt interpretiert (wichtig für alle Tests)

Der EA kann parallel auf mehreren Charts laufen. Damit du nicht aus Versehen die Logs/DB eines anderen Charts interpretierst, musst du bei jedem Test sicherstellen, dass Log-Datei, DB-Datei und (falls aktiv) Discord-Nachrichten zu **dem Chart** passen, den du gerade testest.

Konkret bedeutet das:

- **DB-Datei muss Symbol/TF enthalten:** Wenn du auf EURUSD H1 testest, muss die angelegte Datei **DowHowState_EURUSD_H1.sqlite** heißen (siehe **CDBService::DefaultFile()**).
- **Logs müssen den Kontext zeigen:** In der Log-Datei muss bei Debug-Ausgaben (z.B. **CDBService::PrintData()**) die Zeile **DB file: ...DowHowState_EURUSD_H1.sqlite** auftauchen. Das ist dein „Beweis“, dass du gerade die richtige Instanz anschaust.
- **Discord-Messages müssen Symbol/TF tragen:** Bei Line-Drag-Updates und Send-Signals steht Symbol und TF im Text (z.B. „EURUSD H1 Trade ...“). Das muss zum Chart passen.
- **Wenn etwas nicht passt:** Du liest sehr wahrscheinlich Logs/DB einer anderen Chart-Instanz oder hast mehrere Instanzen laufen. Dann: erst alle EAs vom Chart entfernen, Logs leeren, neu starten.

Praktischer Ablauf zum „Logs passen zu meinem Test“-Check:

- 1 Nur **ein** Chart öffnen (für den Start). EA anhängen.
- 2 Im Terminal → **Experts:** nach „EA Startet OnInit()“ suchen.
- 3 InpDebug=true setzen → EA neu starten → in **DowHowSignalService.log** muss „DB file: ...DowHowState___.sqlite“ erscheinen.
- 4 Im Ordner **MQL5/Files** prüfen, ob genau diese SQLite-Datei existiert und Zeitstempel zur letzten Ausführung passt.

Testkatalog

Hinweis: Viele Tests sind so formuliert, dass sie auch dann funktionieren, wenn du gerade keine echten Trades eröffnest. Der Fokus liegt auf UI/Objekten, DB-Status und Discord-Notifications.

A) Start/Init/Deinit (Lebenszyklus)

INIT-01 — OnInit – erfolgreicher Startpfad

Was wird getestet?	Der komplette Startpfad läuft ohne INIT_FAILED durch: Logger initialisiert, DB init, Router/Discord init, Restore aus DB, UI-Panels/Buttons/Linien werden erzeugt.
Code-Referenzen	<pre>Trade Assistent V2.013.mq5::OnInit() logger.mqh::CLogger::SetLogFileName/SetMethod/Add db_service.mqh::CDBService::Init() webhook_router.mqh::CWebhookRouter::Init/Add/Validate discord_client.mqh::CDiscordClient::Init() trade_manager.mqh::CTradeManager::Init() ui_manager.mqh::CUIManager::Init/RestoreTradeLinesFromDB createEntryAndSLLinien(), TA_Controllers_Init(), g_tp.Create/TP_RebuildRows()</pre>
Voraussetzungen	<ul style="list-style-type: none">Nur 1 Chart geöffnet (für eindeutige Logs).WebRequest für Discord-Domains freigegeben (falls Discord aktiv getestet wird).Input-Webhooks gesetzt oder InpRequireKnownSymbol=false (je nach Test).
Schritte	<ol style="list-style-type: none">EA an Chart anhängen.Experts-Tab + DowHowSignalService.log prüfen.Visuell prüfen: Entry/SL Buttons + HL-Linien vorhanden, Panel links (TP_) vorhanden.
Erwartet	<ul style="list-style-type: none">Im Log: „EA Startet OnInit()“ und danach keine INIT_FAILED-typischen Abbrüche.DB-Datei im Ordner MQL5/Files vorhanden: DowHowState__.sqlite.Keine Spam-Errors (ObjectCreate fails, DatabaseOpen fails, WebRequest denied).
Hinweise / Fehlerbilder	<p>Fail-Indikator: EA verschwindet direkt wieder vom Chart oder Journal zeigt INIT_FAILED.</p> <ul style="list-style-type: none">Typisch bei Discord: „WebRequest is not allowed“ → dann Domain whitelisten.Typisch bei DB: „DatabaseOpen failed“ → Pfad/Permissions prüfen.
Ergebnis	[] PASS [] FAIL Notizen: _____

INIT-02 — OnInit – DB Restore wird ausgeführt

Was wird getestet?	Beim Start wird der vorherige Zustand aus SQLite geladen (Meta + Positionen + Linienpreise).
Code-Referenzen	<pre>Trade Assistent V2.013.mq5::OnInit() → g_TradeMgr.RestoreFromDB(_Symbol,_Period) db_service.mqh::CDBService::PrintData(), LoadPositions(), GetMetaInt/SetMetaInt ui_manager.mqh::CUIManager::RestoreTradeLinesFromDB()</pre>
Voraussetzungen	<ul style="list-style-type: none">InpDebug=true (damit DB.PrintData() Log-Ausgaben schreibt).Einmal vorher mindestens 1 Position/Line gespeichert (z.B. durch Drag/Send).
Schritte	<ol style="list-style-type: none">EA starten, mindestens 1 Preisänderung per Drag ausführen (Entry oder SL oder TradePosLine).EA vom Chart entfernen.EA erneut anhängen.DowHowSignalService.log prüfen (DB file / META / POS Einträge).Visuell: Linien/Tags sollten wieder erscheinen und am gespeicherten Preis liegen.

Erwartet	<ul style="list-style-type: none"> Log zeigt: „DB file: ...DowHowState__.sqlite“. Log zeigt META- und POSITIONS-Zeilen (Anzahl >= 1, je nach vorherigem Zustand). Nach Restart: TradePosLines/Tags werden wieder hergestellt (kein „Reset auf Default“).
Hinweise / Fehlerbilder	<ul style="list-style-type: none"> Wenn META/POSITIONS immer 0 sind: Prüfen ob überhaupt gespeichert wird (Drag-Finalize ruft SaveLinePrices). Wenn falsches Symbol/TF: du hast mehrere DB-Dateien; prüfe Dateinamen.
Ergebnis	[] PASS [] FAIL Notizen: _____

DEINIT-01 — OnDeinit – vollständiges Cleanup

Was wird getestet?	Beim Entfernen des EA werden registrierte UI-Objekte gelöscht, Panels zerstört, Registry zurückgesetzt und DB geschlossen.
Code-Referenzen	<pre>Trade_Assistent V2.013.mq5::OnDeinit() ui_registry.mqh::UI_Reg_DeleteAll() trade_pos_line_registry.mqh::TradePosLines_Reset() trades_panel_class.mqh::g_tp.Destroy() db_service.mqh::CDBService::Close()</pre>
Voraussetzungen	<ul style="list-style-type: none"> EA läuft und hat UI-Objekte erstellt (INIT-01 bestanden).
Schritte	<ol style="list-style-type: none"> EA vom Chart entfernen. Im Log prüfen: Zeile „OnDeinit(reason=...): UI Objekte gelöscht=...“ Chart visuell prüfen: Entry/SL/TP_-Objekte sollten verschwunden sein (keine Reste).
Erwartet	<ul style="list-style-type: none"> UI_Reg_DeleteAll() entfernt die registrierten Objekte (deleted > 0 in typischen Runs). Keine „dangling tags“ (Tags ohne Linien) verbleiben im Chart. DB wird ohne Fehler geschlossen (keine DatabaseFinalize-Leaks sichtbar).
Hinweise / Fehlerbilder	Wenn Objekte bleiben: sind sie evtl. nicht im Registry registriert → Registry/Creation prüfen.
Ergebnis	[] PASS [] FAIL Notizen: _____

B) Log/DB/Discord-Konsistenz (dein "Logs spiegeln Inputs"-Punkt)

CONS-01 — DB-Dateiname passt zu Chart-Symbol/TF

Was wird getestet?	Sicherstellen, dass du beim Testen nicht aus Versehen eine andere EA-Instanz/DB ausliest. Der Dateiname ist der einfachste harte Beweis.
Code-Referenzen	<pre>db_service.mqh::CDBService::DefaultFile() db_service.mqh::CDBService::Init() / PrintData() (loggt „DB file: ...“)</pre>
Voraussetzungen	<ul style="list-style-type: none"> InpDebug=true (für PrintData-Log).
Schritte	<ol style="list-style-type: none"> Chart auf ein eindeutiges Symbol/TF stellen (z.B. EURUSD H1). EA starten (OnInit). Im Ordner MQL5/Files prüfen: existiert DowHowState_EURUSD_H1.sqlite? In DowHowSignalService.log nach „DB file:“ suchen.
Erwartet	<ul style="list-style-type: none"> DB-Dateiname enthält genau das Symbol und TF des Charts. Log-Zeile „DB file:“ zeigt denselben Dateinamen.
Hinweise / Fehlerbilder	Wenn Datei anders heißt: du testest nicht das, was du glaubst (falsches Chart, mehrere Instanzen, falsches Symbol alias).

Ergebnis	[] PASS [] FAIL Notizen: _____
----------	----------------------------------

CONS-02 — Discord-Update nach Linien-Drag trägt richtigen Kontext

Was wird getestet?	Wenn du eine TradePosLine (Entry/SL einer Position) verschiebst, muss die Discord-Nachricht Symbol/TF/Trade/Pos enthalten, die zu deiner Testaktion passen.
Code-Referenzen	<pre>CTradePosLineDragController.mqh::Finalize() trade_manager.mqh::CTradeManager::SaveLinePrices() discord_client.mqh::CDiscordClient::SendMessage()</pre>
Voraussetzungen	<ul style="list-style-type: none"> • Discord-Webhook aktiv (g_Discord.Init erfolgreich). • Mindestens 1 TradePosLine existiert (z.B. LONG Trade 1 Pos 1 Entry/SL Linie).
Schritte	<ol style="list-style-type: none"> 1 Eine TradePosLine (z.B. LONG Trade 1 Pos 1 Entry) anklicken und deutlich verschieben. 2 Maus loslassen (Finalize muss feuern). 3 Discord-Channel prüfen (UPDATE Nachricht).
Erwartet	<ul style="list-style-type: none"> • Message enthält: Symbol (z.B. EURUSD) und TF (z.B. H1) und Trade/Pos Nummern. • Preisänderung wird korrekt angezeigt (old -> new). • Nachricht kommt nur, wenn Preis sich wirklich geändert hat (Toleranz pt*0.25).
Hinweise / Fehlerbilder	<ul style="list-style-type: none"> • Wenn jede kleinste Bewegung spammt: changed-Logik/pt Toleranz prüfen. • Wenn keine Nachricht: g_Discord.Init fehlgeschlagen oder WebRequest nicht erlaubt.
Ergebnis	[] PASS [] FAIL Notizen: _____

C) TradePosLines Objektmodell (CTradePosLine + Registry + DragController)

TPL-01 — CTradePosLine::Create erzeugt Linie + Tag (lesbar im Chart)

Was wird getestet?	Eine TradePosLine besteht aus HL-Linie + Tag-Label. Tag muss sichtbar im Chart liegen (nicht rechts außerhalb).
Code-Referenzen	<code>CTradePosLine.mqh::Create()</code> , <code>CreateTagIfNeeded()</code> , <code>SyncTagToLine()</code> <code>CTradePosLine.mqh::SetTagShiftBars()</code> , <code>SetTagOffsetPx()</code> , <code>SetTagCorner()</code>
Voraussetzungen	<ul style="list-style-type: none">• Chart mit genügend Historie (nicht ganz nach rechts gescrollt).• Mindestens 1 TradePosLine wird erzeugt (durch Restore oder manuell über Codepfad).
Schritte	<ol style="list-style-type: none">1 TradePosLine erzeugen lassen (z.B. durch <code>RestoreTradeLinesFromDB</code> oder durch <code>Send/Panel-Funktion</code>).2 Visuell prüfen: Linie vorhanden; Tag-Text vorhanden und lesbar.3 Chart zoomen (rein/raus) und etwas scrollen.
Erwartet	<ul style="list-style-type: none">• Tag bleibt nahe der Linie und bleibt innerhalb der sichtbaren Chartfläche.• Tag klebt an der Linie (<code>SyncTagToLine</code> nutzt <code>ChartTimePriceToXY</code>).
Hinweise / Fehlerbilder	<ul style="list-style-type: none">• Wenn Tag weit rechts „klebt“: <code>TagShiftBars/Offset</code> zu groß oder <code>corner</code> falsch.• Wenn Tag „verschwindet“: <code>ChartTimePriceToXY</code> liefert false (Zeit außerhalb sichtbarer Range).
Ergebnis	[] PASS [] FAIL Notizen: _____

TPL-02 — Linie startet gestrichelt, kann auf durchgezogen wechseln

Was wird getestet?	Neue TradePosLines sind initial gestrichelt (<code>Style=DOT</code>). Später kann auf durchgezogen gewechselt werden (z.B. bei Statuswechsel <code>PENDING→OPEN</code>).
Code-Referenzen	<code>CTradePosLine.mqh::SetDashed()</code> , <code>SetSolid()</code> <code>sonstige_methoden.mqh::TPSLReached()</code> → <code>g_TradeMgr.SetPosLinesSolid(...)</code> (bei <code>Entry Hit</code>)
Voraussetzungen	<ul style="list-style-type: none">• Mindestens 1 TradePosLine existiert.• Optional: Statuswechsel durch Testdaten oder manuell in DB erzwingen.
Schritte	<ol style="list-style-type: none">1 Nach Erstellung prüfen: Linie ist gestrichelt.2 Statuswechsel simulieren: entweder <code>Entry-Hit</code> abwarten oder (für QA) in DB <code>status='OPEN'</code> setzen und <code>Restore</code> laufen lassen.3 Prüfen: Linie wird durchgezogen dargestellt.
Erwartet	<ul style="list-style-type: none">• <code>DOT</code>-Style bei neuen Linien, <code>SOLID</code>-Style nach <code>SetSolid</code>.• Style-Wechsel betrifft die richtige Linie (nicht <code>PR_HL/SL_HL</code> der Basis-Buttons).
Hinweise / Fehlerbilder	• Wenn falsche Linie umgestellt wird: Namensparsing (<code>UI_IsTradePosLine</code> / Registry Lookup) prüfen.
Ergebnis	[] PASS [] FAIL Notizen: _____

TPL-03 — Drag – Tag bewegt sich ohne spürbare Verzögerung

Was wird getestet?	Beim Drag einer TradePosLine muss das Tag nahezu synchron nachlaufen (keine Sekunden-Verzögerung).
Code-Referenzen	<code>CTradePosLineDragController.mqh::OnChartEvent()</code> / <code>OnMouseMove()</code> / <code>ThrottledTagSync()</code> <code>CTradePosLine.mqh::SyncTagToLine()</code>

Voraussetzungen	<ul style="list-style-type: none"> Eine TradePosLine mit Tag ist vorhanden.
Schritte	<ol style="list-style-type: none"> 1 Linie anklicken und langsam ziehen. 2 Dabei auf das Tag achten: folgt es direkt oder „springt“ es hinterher? 3 Linie loslassen.
Erwartet	<ul style="list-style-type: none"> Während Drag: Tag folgt (ThrottledTagSync: max ~30 FPS). Nach Loslassen: Tag sitzt exakt am finalen Preis (Finalize ruft UI_CreateOrUpdateLineTag + Redraw).
Hinweise / Fehlerbilder	<ul style="list-style-type: none"> Wenn es laggt: Throttle-Intervall zu hoch oder ChartRedraw wird zu selten ausgelöst. Wenn es nach Loslassen falsch sitzt: Finalize/SyncTagToLine Fehler oder falscher Name in m_name.
Ergebnis	[] PASS [] FAIL Notizen: _____

TPL-04 — Finalize – DB Save + Discord Update nur bei echter Änderung

Was wird getestet?	Nach Drag-Ende: (1) Tag final updaten, (2) Preise in DB speichern, (3) Discord senden nur wenn Preisänderung > pt*0.25.
Code-Referenzen	<pre>CTradePosLineDragController.mqh::Finalize() trade_manager.mqh::CTradeManager::SaveLinePrices()</pre>
Voraussetzungen	<ul style="list-style-type: none"> Discord aktiv oder alternativ nur DB prüfen. InpDebug=true um DB.PrintData zu nutzen.
Schritte	<ol style="list-style-type: none"> 1 Eine TradePosLine minimal bewegen (< 0.25 Punkt) und loslassen. 2 Dann deutlich bewegen (> 1 Punkt) und loslassen. 3 InpDebug=true: DB.PrintData ausführen (EA restart) oder SQLite prüfen.
Erwartet	<ul style="list-style-type: none"> Bei Mini-Bewegung: keine Discord-UPDATE Nachricht (changed=false). Bei deutlicher Bewegung: Discord-UPDATE Nachricht erscheint. DB enthält den finalen Preis (LoadPositions/PrintData zeigt neuen entry/sl).
Ergebnis	[] PASS [] FAIL Notizen: _____

D) Basis-UI (Entry/SL Gruppe + PR_HL/SL_HL)

BASE-01 — createEntryAndSLLinien – Basisobjekte werden erzeugt und sind bedienbar

Was wird getestet?	Entry/SL Buttons, Eingabefelder und PR_HL/SL_HL Linien existieren nach Start. Linien sind anklick- und dragbar.
Code-Referenzen	<pre>Trade Assistent V2.013.mq5::OnInit() → createEntryAndSLLinien() event.mqh::OnChartEvent(...) (OBJ_DRAG/CLICK) – Basis-Linien-Handling</pre>
Voraussetzungen	-
Schritte	<ol style="list-style-type: none"> 1 EA starten. 2 PR_HL anklicken und verschieben. 3 SL_HL anklicken und verschieben.
Erwartet	<ul style="list-style-type: none"> Beide HL-Linien lassen sich ziehen (OBJPROP_SELECTABLE/OBJPROP_SELECTED korrekt). Die zugehörigen UI-Elemente (Button-Text/Preis) werden aktualisiert.

Hinweise / Fehlerbilder	• Wenn SL_HL nicht anklickbar: prüfen ob OBJPROP_SELECTABLE/OBJPROP_HIDDEN gesetzt ist.
Ergebnis	[] PASS [] FAIL Notizen: _____

BASE-02 — Drag Basislinien synchronisiert Button-Text und interne Preise

Was wird getestet?	Beim Verschieben von PR_HL/SL_HL werden Entry_Price/SL_Price aktualisiert und die UI-Texte (Preis/Lot/Buy-Sell) werden konsistent angepasst.
Code-Referenzen	event.mqh::MouseMove-States / UpdateEntrySLText (je nach Implementierung) trade_manager.mqh::calcLots() (falls Lot berechnet wird) Trade Assistant V2.013.mq5: Variablen Entry_Price, SL_Price
Voraussetzungen	• Lot-/Text-Update muss aktiv sein (kein Debug-Stub).
Schritte	1 PR_HL nach oben/unten ziehen. 2 SL_HL so verschieben, dass SL > Entry wird und wieder zurück. 3 Buttontexte prüfen (Richtungswechsel Buy/Sell, Lot, Distanz).
Erwartet	• Preis im Text entspricht dem Linienpreis (ObjectGetDouble OBJPROP_PRICE). • Bei SL > Entry: Richtungslogik dreht wie vorgesehen. • Keine NaN/0 Werte, keine falschen Dezimalstellen.
Hinweise / Fehlerbilder	• Wenn Texte nicht stimmen: Reihenfolge der Updates (erst Preis lesen, dann Text setzen) prüfen.
Ergebnis	[] PASS [] FAIL Notizen: _____

E) Trades Panel (TP_*)

TP-01 — Panel Create + RebuildRows erzeugt konsistente UI

Was wird getestet?	Das neue Panel (g_tp) wird erstellt und die Rows werden anhand DB/Cache sauber aufgebaut.
Code-Referenzen	<code>Trade Assistant V2.013.mq5::OnInit() → g_tp.Create(), TP_RebuildRows(), g_tp.RebuildRows()</code> <code>trades_panel_class.mqh::Create(), RebuildRows(), Destroy()</code> <code>trades_panel.mqh::TP_DeleteByPrefix(), TP_RebuildRows()</code> (falls Wrapper genutzt)
Voraussetzungen	-
Schritte	<ol style="list-style-type: none">1 EA starten.2 Visuell: Panel links sichtbar, keine überlappenden Rows.3 Mehrere Positionen erzeugen (SendDraft) und danach RebuildRows auslösen (z.B. durch Restart).
Erwartet	<ul style="list-style-type: none">• Rows entsprechen DB-Positions: pro Position eine Zeile mit Buttons (Cancel/Stop).• Bei 0 Positionen: Panel zeigt leeren Zustand ohne Fehlermeldungen.
Hinweise / Fehlerbilder	Wenn Rows doppelt erscheinen: Cleanup vor Create/Rebuild (TP_DeleteByPrefix) prüfen.
Ergebnis	[] PASS [] FAIL Notizen: _____

TP-02 — Row-Buttons: Cancel/Stop lösen TradeManager-Aktionen aus

Was wird getestet?	Klick auf Cancel/Stop in einer Row führt zu HandlePositionAction / CancelTrade und aktualisiert UI + DB + ggf. Discord.
Code-Referenzen	<code>trade_manager.mqh::HandlePositionAction(), CancelTrade()</code> <code>event.mqh::OnChartEvent - Button click routing</code> <code>db_service.mqh::UpsertPosition()</code> (status update), <code>LoadPositions()</code>
Voraussetzungen	<ul style="list-style-type: none">• Mindestens 1 Position existiert (was_sent=1 oder testweise).• Discord optional aktiv.
Schritte	<ol style="list-style-type: none">1 Im Panel eine Row auswählen.2 Cancel klicken → prüfen was passiert (Status / Linien löschen).3 Stop klicken → prüfen was passiert (Status / Linien löschen).4 EA restart → DB.PrintData prüfen (status verändert?).
Erwartet	<ul style="list-style-type: none">• Aktion ändert Status in DB (z.B. CLOSED oder CLOSED_SL je nach Action).• TradePosLines werden gelöscht (DeletePosLines).• Panel aktualisiert sich (Row verschwindet oder Status ändert sich).
Ergebnis	[] PASS [] FAIL Notizen: _____

F) Business Rules (max 4 simultane Positionen, Re-Use nach Close)

RULE-01 — Max 4 gleichzeitig pro Richtung/Trade: 5. wird verhindert

Was wird getestet?	Es dürfen maximal 4 gleichzeitige aktive Positionen existieren. Die 5. aktive Erzeugung muss blockieren.
Code-Referenzen	<code>db_service.mqh::CountActivePositions(), GetNextPosNo(), LoadPositions()</code> <code>trade_manager.mqh::SendSignalDraft() / CreateDraft()</code> (PosNo Vergabe)
Voraussetzungen	<ul style="list-style-type: none">• Panel/Send-Funktion erzeugt Positionen in DB.

Schritte	1 4 Positionen nacheinander erzeugen (z.B. LONG Trade 1 Pos1..Pos4). 2 Versuchen, eine 5. Position zu erzeugen. 3 DB.PrintData prüfen.
Erwartet	<ul style="list-style-type: none"> Position 5 wird nicht als aktiv angelegt (kein zusätzlicher POS-Eintrag als aktiv). UI zeigt keine 5. Row für aktive Positionen. Ggf. Fehlermeldung/Return-Code (SEND_ERR_MAXPOS).
Hinweise / Fehlerbilder	Wenn doch 5 existiert: CountActivePositions Filter auf Status/was_sent/is_pending prüfen.
Ergebnis	[] PASS [] FAIL Notizen: _____

RULE-02 — Nach Schließen darf wieder eine neue Position entstehen (trotz früherem Pos5)

Was wird getestet?	Wenn eine Position geschlossen wurde, darf wieder eine neue Position eröffnet werden, solange nicht >4 gleichzeitig aktiv sind. PosNo kann dabei >4 sein (historisch).
Code-Referenzen	db_service.mqh::CountActivePositions() (nur aktive zählen) db_service.mqh::GetNextPosNo() (MAX(pos_no)+1 historisch)
Voraussetzungen	<ul style="list-style-type: none"> Mindestens 4 Positionen existieren, mindestens 1 davon geschlossen.
Schritte	1 4 Positionen anlegen. 2 Eine Position schließen (Cancel/Stop oder status auf CLOSED setzen). 3 Neue Position anlegen. 4 DB.PrintData prüfen: Anzahl aktiver Positionen <=4.
Erwartet	<ul style="list-style-type: none"> Neue Position wird erlaubt (weil aktive < 4). PosNo kann 5 sein (historisch), aber gleichzeitig aktiv bleiben max. 4.
Hinweise / Fehlerbilder	Das ist kein Bug: PosNo ist eine laufende Nummer, nicht „Slot 1..4“. Relevant ist nur die Anzahl aktiver Positionen.
Ergebnis	[] PASS [] FAIL Notizen: _____

G) Negativ-/Fehlerpfad-Tests

ERR-01 — Discord WebRequest nicht erlaubt → sauberer Fehler

Was wird getestet?	Wenn WebRequest nicht freigeschaltet ist, muss der EA den Fehler klar loggen und (je nach Design) Init abbrechen oder Discord deaktivieren.
Code-Referenzen	<code>discord_client.mqh::Init()</code> , <code>TestConnection()</code> , <code>SendMessage()</code> <code>MT5 WebRequest Permissions</code>
Voraussetzungen	<ul style="list-style-type: none">• In MT5: WebRequest-Domains NICHT freigeben.
Schritte	<ol style="list-style-type: none">1 EA starten (Discord aktiv).2 Journal/Experts prüfen.
Erwartet	<ul style="list-style-type: none">• Klarer Hinweis in Logs (WebRequest denied / error code 4014).• Kein stilles „Nichts passiert“ ohne Log.
Ergebnis	[] PASS [] FAIL Notizen: _____

ERR-02 — Unbekanntes Symbol bei InpRequireKnownSymbol=true → Init blockiert

Was wird getestet?	Router-Validierung muss verhindern, dass ein unbekanntes Symbol ohne Webhook-Mapping läuft (um falsche Discord-Ziele zu vermeiden).
Code-Referenzen	<code>webhook_router.mqh::CWebhookRouter::Validate()</code> <code>Trade Assistant V2.013.mq5::OnInit() → g_router.Validate()</code>
Voraussetzungen	<ul style="list-style-type: none">• <code>InpRequireKnownSymbol=true</code>• Chart auf ein Symbol ohne Mapping stellen (z.B. exotisches Symbol).
Schritte	<ol style="list-style-type: none">1 EA starten.2 Journal/Experts prüfen, ob <code>INIT_FAILED</code> kommt.
Erwartet	<ul style="list-style-type: none">• Init bricht mit klarer Log-Meldung ab (fehlendes Mapping).• Kein späteres Senden an falsche Webhooks.
Ergebnis	[] PASS [] FAIL Notizen: _____

Abschluss / Sign-off

Wenn alle Tests PASS sind, ist der aktuelle Stand regressions-sicher für: Start/Restore/Cleanup, TradePosLines-Objektmodell, Drag-Sync, DB-Persistenz und Discord-Updates. Empfehlung: Logs/DB-Artefakte (Log-Datei + SQLite) zusammen mit Screenshots pro Fail-Fall sammeln.

Sign-off

Tester: _____ Datum: _____ Build: _____