

# MQL5 Trading Assistant - Qualitätssicherung Checkliste

## Kritische Variablen & Initialization

### Global Variables Validation

- ☐ `tradenummer` - Prüfe auf negative Werte und Overflow
- ☐ `last_trade_nummer` - Konsistenz mit gespeicherten Settings
- ☐ `Entry_Price`, `TP_Price`, `SL_Price` - Null/negative Werte abfangen
- ☐ `CurrentAskPrice`, `CurrentBidPrice` - Symbol-spezifische Validierung
- ☐ `is_long_trade`, `is_sell_trade` - Mutex-Verhalten sicherstellen
- ☐ `working_webhook_*` URLs - Format und Erreichbarkeit prüfen
- ☐ `working_ShowTPButton` - UI-Konsistenz bei false

### Settings Structure

- ☐ `g_Settings` - Alle char arrays auf Buffer Overflow prüfen
  - ☐ Webhook URLs max 256 Zeichen Limit
  - ☐ Settings Version String korrekt konvertiert
  - ☐ Default-Werte für alle Settings definiert
- 

## Function Parameter Validation

### calcLots() Function

- ☐ `slDistance <= 0` - Negativer/Null SL-Distance
- ☐ `ticksize == 0` - Division durch Null
- ☐ `tickvalue == 0` - Ungültige Symbol-Daten
- ☐ `lotstep == 0` - Broker-Parameter fehlen
- ☐ `riskMoney <= 0` - Ungültiges Risiko-Kapital
- ☐ `lots < minlot` - Unter Broker-Minimum
- ☐ `lots > maxlot` - Über Broker-Maximum
- ☐ Normalisierung mit korrekten Dezimalstellen

### Price Validation Functions

- ☐ `Get_Price_d()` - Object existiert und hat gültigen Preis
  - ☐ `Get_Price_s()` - String-Konvertierung mit korrekten Digits
  - ☐ `ObjectGetDouble()` Return-Werte auf 0.0 prüfen
  - ☐ `NormalizeDouble()` mit `_Digits` Konsistenz
- 

## UI Object Management

### Button Creation

- ☐ `createButton()` - Alle Parameter auf Gültigkeit prüfen
- ☐ `xD`, `yD` innerhalb Chart-Grenzen
- ☐ `xS`, `yS` positive Werte
- ☐ `ObjectCreate()` Erfolg prüfen
- ☐ Font-Größe zwischen 6-72
- ☐ Farb-Werte gültig (nicht clrNONE wenn nicht gewollt)

## Line Creation

- ☐ `createHL()` und `CreateTPSLLines()` Object-Creation Erfolg
- ☐ `price1` innerhalb Symbol-Range
- ☐ `time1` gültiger datetime Wert
- ☐ Style und Color Parameter validieren

## Object Visibility Management

- ☐ `OBJ_NO_PERIODS` vs `OBJ_ALL_PERIODS` Konsistenz
  - ☐ `working_ShowTPButton` bedingte Sichtbarkeit
  - ☐ UI-State nach Settings-Laden konsistent
- 



## Discord Integration

### Webhook Validation

- ☐ `get_discord_webhook()` - Nie empty string zurückgeben
- ☐ URL Format validierung (<https://discord.com/...>)
- ☐ Fallback zu `discord_webhook_test` funktioniert
- ☐ Timeframe-basierte Webhook-Auswahl korrekt

### Message Sending

- ☐ `SendDiscordMessage()` - WebRequest Error Handling
- ☐ HTTP Status Codes 200/204 als Erfolg
- ☐ Rate Limiting (429) behandeln
- ☐ `EscapeJSON()` für alle Sonderzeichen
- ☐ Message-Länge Limits (2000 Zeichen Discord)

### Screenshot Function

- ☐ `SendScreenShot()` - File Creation Erfolg prüfen
  - ☐ Chart Exists und Symbol Match
  - ☐ File-Größe > 0 vor Upload
  - ☐ File-Cleanup nach Upload
  - ☐ Error Handling bei File-Operations
- 



## Settings & Persistence

## Save Settings

- ☐ `SaveSettings()` - File Handle Validation
- ☐ Struct-Size Konsistenz zwischen Save/Load
- ☐ String zu CharArray Konvertierung vollständig
- ☐ File Path Permissions prüfen
- ☐ Disk Space verfügbar

## Load Settings

- ☐ `LoadSettings()` - File Existence Check
  - ☐ Version Compatibility prüfen
  - ☐ CharArray zu String korrekt
  - ☐ Corrupted File Detection
  - ☐ Default Values bei Load Failure
  - ☐ TradeInfo Array Bounds
- 

## Event Handling

### OnChartEvent

- ☐ Mouse Coordinates innerhalb Chart
- ☐ Object State Changes validieren
- ☐ `movingState_*` Mutex Behaviour
- ☐ Button Bounds Detection korrekt
- ☐ Parameter Hash Berechnung konsistent

### OnTick Performance

- ☐ Price Cache Update Logic
  - ☐ `need_label_update_*` Flags korrekt gesetzt
  - ☐ Tick-Rate bei high-frequency Symbolen
  - ☐ Memory Usage bei Labels Update
- 

## Trading Logic

### Trade Validation

- ☐ `DiscordSend()` - Entry vs Current Price Logic
- ☐ `tradenummer > last_trade_nummer` Validation
- ☐ Duplicate Trade Prevention
- ☐ Order Type Selection (Buy/Sell Stop)
- ☐ Lot Size Validation vor Order Send

## TP/SL Monitoring

- ☐ `TPSLReached()` - Price Comparison Accuracy
- ☐ `HitEntryPrice*` State Management
- ☐ Order Execution vor Entry-Hit
- ☐ Slippage bei Market Orders
- ☐ Position Tracking Consistency

## Order Management

- ☐ `DeleteBuyStopOrderForCurrentChart()` - Symbol Filter
  - ☐ Order Type Filter korrekt
  - ☐ Multiple Orders handling
  - ☐ Partial Fills berücksichtigen
- 

## Error Handling

### Error Code System

- ☐ Alle `TRADE_ERROR_CODE` Enums abgedeckt
- ☐ `GetErrorMessage()` für alle Codes
- ☐ `ShowTradeError()` Sound + Logging
- ☐ Error Recovery Mechanisms
- ☐ User-friendly deutsche Meldungen

### Exception Handling

- ☐ `GetLastError()` nach kritischen Operationen
  - ☐ File Operations Error Codes
  - ☐ Network Error Handling (Discord)
  - ☐ Symbol Information Failures
  - ☐ Memory Allocation Failures
- 

## Security & Validation

### Input Sanitization

- ☐ Trade Number Input Range (0-9999999)
- ☐ Sabio Price Fields SQL Injection safe
- ☐ URL Webhook Format Validation
- ☐ File Path Traversal Prevention

### Business Logic Validation

- ☐ Risk Management Limits eingehalten
  - ☐ Maximum Positions per Symbol
  - ☐ Daily Loss Limits
  - ☐ Position Size vs Account Equity
-



# Performance & Memory

## Memory Management

- ☐ `deleteObjects()` alle Objects erfasst
- ☐ `DeleteAllObjectsByPrefix()` Memory Leaks
- ☐ Global Variables Cleanup in OnDeinit
- ☐ String Buffer Overflows
- ☐ Array Bounds bei TradeInfo

## Performance Optimization

- ☐ `OnTick()` CPU Usage minimiert
- ☐ Unnötige `ChartRedraw()` Calls vermeiden
- ☐ Cache Mechanismen funktional
- ☐ Label Updates nur bei Änderungen



## Integration & Compatibility

### MT5 Platform

- ☐ Symbol-spezifische `_Digits` Verwendung
- ☐ Broker-spezifische Lot Steps
- ☐ Timeframe Compatibility
- ☐ Market Hours Berücksichtigung

### Multi-Chart Support

- ☐ Chart ID Unique Identifiers
- ☐ Settings per Chart/Symbol
- ☐ Global Variable Conflicts vermeiden
- ☐ Cross-Chart Communication



## Testing Scenarios

### Edge Cases

- ☐ Null/Empty Symbol Testing
- ☐ Market Closed Scenarios
- ☐ Internet Connection Loss
- ☐ Broker Disconnection
- ☐ Extreme Price Movements
- ☐ Multiple EA Instances

### User Input Validation

- ☐ Extreme Zahlen-Eingaben
  - ☐ Ungültige Zeichen in Feldern
  - ☐ Copy/Paste in Edit Fields
  - ☐ Sehr lange Webhook URLs
  - ☐ Sonderzeichen in Nachrichten
- 

## **Deployment Checklist**

### **Pre-Release**

- ☐ Alle Input Parameter dokumentiert
- ☐ Default Values produktionstauglich
- ☐ Debug Print Statements entfernt/reduziert
- ☐ Version Number aktualisiert
- ☐ Backup/Recovery tested

### **Post-Release Monitoring**

- ☐ Error Log Monitoring Setup
  - ☐ Performance Metrics Collection
  - ☐ User Feedback Integration
  - ☐ Update Mechanism funktional
- 

## **Automatisierte Tests**

### **Unit Tests (zu implementieren)**

- ☐ calcLots() mit verschiedenen Szenarien
- ☐ Price Validation Functions
- ☐ Discord Message Formatting
- ☐ Settings Save/Load Roundtrip
- ☐ Error Message Generation

### **Integration Tests**

- ☐ Discord API Connectivity
  - ☐ MT5 Order Placement
  - ☐ File System Operations
  - ☐ UI Object Interactions
  - ☐ Cross-Function Dependencies
- 

## **Dokumentation**





### **Code Documentation**

- ☐ Alle Functions haben Header Comments
- ☐ Parameter Descriptions vollständig
- ☐ Return Value Documentation
- ☐ Error Conditions dokumentiert
- ☐ Version History aktuell

## User Documentation

- ☐ Setup Instructions
  - ☐ Parameter Explanations
  - ☐ Troubleshooting Guide
  - ☐ FAQ Section
  - ☐ Contact Information
- 

## Status Legende:

-  Vollständig getestet und validiert
-  Teilweise implementiert, needs review
-  Kritischer Fehler identifiziert
-  Noch zu implementieren/testen