HCX Elite University TCP Bridge Integration Manual

1. Overview

This document explains how to integrate the **Raw TCP Bridge Layer** (`TCPRawBridge.mqh`) into your main Expert Advisor (EA) for production use. The bridge enables reliable, low-level communication between MetaTrader 5 and an external Python or VPS-based system via persistent TCP sockets.

The bridge uses **Winsock (ws2_32.dll)** directly, bypassing MQL5 abstractions to achieve maximum compatibility and control. It uses a **4-byte length-prefixed JSON protocol** for efficient message exchange.

---

2. File Structure

Ensure the following structure inside your MetaTrader 5 data directory:

```
MQL5/
├── Experts/
│   └── test_tcp_bridge.mq5
│
├── Include/
│   ├── TCPRawBridge.mqh
│   └── WinsockImports.mqh
```

File Roles

File	Purpose
WinsockImports.mqh	Declares raw Winsock DLL functions (`socket`, `send`, `recv`, etc.). Must remain simple and unguarded.
TCPRawBridge.mqh	Implements TCP connection logic, JSON framing, reconnection, signal tracking, and ping/pong.
test_tcp_bridge.mq5	Example EA demonstrating integration with the TCP bridge and mock trading signal transmission.

---

3. Setup Steps

Step 1: File Placement

Copy both header files to the `MQL5\Include\` directory and your EA (e.g., `test_tcp_bridge.mq5`) to `MQL5\Experts\`.

Step 2: EA Inclusion

At the top of your EA, include the bridge header (you can use the examplemq5 to guide you):

```
#include <TCPRawBridge.mqh>
```

---

4. Bridge Architecture

The TCP bridge provides: - Raw Winsock socket management - Persistent TCP connection to external server - JSON message handling - Exponential reconnection with backoff - Heartbeat (ping/pong) - Multi-symbol signal tracking (`g_signals[]`)

Message Format

Each message consists of:

```
[4-byte length header][UTF-8 JSON payload]
```

Example payloads:

```
{"type":"signal","client_msg_id":"2025-10-31-
001","action":"BUY","symbol":"EURUSD","price":1.0765,"allow_close":true}
{"type":"signal","client_msg_id":"2025-10-31-
002","action":"CLOSE","open_signal_id":12,"price":1.0780}
```

---

5. EA Integration

Initialization

In `OnInit()`:

```
if(!TCP_Init(Inp_BridgeHost, Inp_BridgePort))
   Print("Initial connect failed; will retry in OnTimer.");

EventSetTimer(Inp_PollIntervalSec);
```

Polling and Heartbeat (OnTimer)

```
void OnTimer()
{
   if(!TCP_IsConnected()) TCP_ReconnectIfNeeded();
   TCP_Poll();
}
```

Sending a Signal

```
string cid = SignalOpen(_Symbol, "BUY", Ask, true);
```

Registers a signal locally and sends it to the bridge.

Closing a Signal

```
SignalClose(server_signal_id, Bid);
```

Closes a signal previously confirmed by the server.

---

6. Example EA: test_tcp_bridge.mq5

The included example EA demonstrates a working implementation: - Connects to bridge at startup - Sends one signal per test symbol - Periodically pings the bridge - Optionally auto-closes open signals after a duration

Logs appear in the **Experts** tab:

```
[TCP] Connected to 127.0.0.1:31415
[SignalOpen] Sent client_msg_id=2025-10-31-001 symbol=EURUSD
[TCP] Confirmed client=2025-10-31-001 -> server_id=12
[TCP] Closed confirmation received for open_client_id=2025-10-31-001
```

---

7. Production Integration

When moving to production:

1. Include `#include <TCPRawBridge.mqh>` at the top of your EA.
2. Initialize the connection in `OnInit()`.
3. Add `TCP_Poll()` and `TCP_ReconnectIfNeeded()` to your `OnTimer()`.
4. Use `SignalOpen()` and `SignalClose()` in your EA's trading logic.

Example:

```
if(buy_condition)
{
   OrderSend(...);
   SignalOpen(_Symbol, "BUY", Ask, true);
}

if(close_condition)
{
   int open_id = g_signals[0].server_signal_id;
   SignalClose(open_id, Bid);
}
```

---

8. Testing Procedure

1. Run your Python bridge server locally on port `31415`.
2. Attach `test_tcp_bridge.mq5` to a chart.
3. Enable **DLL Imports** and **Algo Trading**.
4. Observe logs for connection and signal confirmations.

---

9. Production Notes

Aspect	Recommendation
Port	Use a dedicated port (e.g., 31415). Avoid common ones.
Firewall	Allow inbound/outbound access on the chosen port.
Timeouts	Default `recv()` timeout is 50ms. Adjust if needed.
Logging	All logs are written to the Experts tab. Mirror them if desired.
Persistence	To persist signals across restarts, serialize `g_signals[]` to file.

---

10. Quick Integration Checklist

Step	Task	Done
1	Copy header files into `MQL5/Include/`	☐
2	Include `<TCPRawBridge.mqh>` in EA	☐
3	Call `TCP_Init()` in `OnInit()`	☐
4	Add `TCP_Poll()` + `TCP_ReconnectIfNeeded()` in `OnTimer()`	☐
5	Use `SignalOpen()` and `SignalClose()` in EA logic	☐
6	Enable DLL imports	☐
7	Test with Python bridge	☐
8	Verify logs and confirmations	☐

---

11. Minimal Production Template

```
#property strict
#include <TCPRawBridge.mqh>

input string Host = "127.0.0.1";
input int Port = 31415;

int OnInit()
{
```

```
   TCP_Init(Host, Port);
   EventSetTimer(1);
   return(INIT_SUCCEEDED);
}

void OnTimer()
{
   TCP_ReconnectIfNeeded();
   TCP_Poll();
}

void OnTick()
{
   if(condition_to_buy)
       SignalOpen(_Symbol, "BUY", Ask, true);

   if(condition_to_close)
       SignalClose(g_signals[0].server_signal_id, Bid);
}

void OnDeinit(const int reason)
{
   EventKillTimer();
   TCP_Close();
}
```

---

12. Conclusion

This integration provides a stable, extensible communication layer between your MT5 trading systems and external AI or risk management infrastructure. By embedding the raw TCP bridge, you ensure resilience, minimal latency, and precise control over how your EA exchanges trade signals and confirmations with your backend.