# White Paper: Architectural Decision for AI-Powered Trading System in MetaTrader 4 (MT4)

\*\*Author:\*\* Amu Hlongwane, PHD Quantitative Analytics

\*\*Date:\*\* [Insert Date]

## Abstract

This white paper evaluates different architectural options for integrating an AI-powered trading system with MetaTrader 4 (MT4). The three primary approaches considered were:  
1. \*\*Java JeroMQ\*\* – A pure Java implementation of ZeroMQ enabling AI-based trading models.  
2. \*\*MQL4 with `libzmq.dll`\*\* – A direct integration of ZeroMQ with MetaTrader 4 using a Windows-based DLL.  
3. \*\*Windows API (`user32.dll`)\*\* – An approach leveraging Windows inter-process communication (IPC).  
  
We analyze each approach based on performance, reliability, security, efficiency, and maintainability to justify the final design choice.

## 1. Introduction

### 1.1 Background

The global financial markets are increasingly adopting AI-powered trading systems to enhance decision-making processes, improve execution efficiency, and minimize trading risks. MetaTrader 4 (MT4), a widely used trading platform, lacks native AI integration. To enable seamless AI-based trading strategies, an efficient communication layer must be established between MT4 and an external AI server.

### 1.2 Problem Statement

Traditional trading automation in MT4 relies on predefined rule-based strategies. However, machine learning and reinforcement learning models provide superior predictive capabilities. The challenge is selecting the most suitable architecture to enable real-time, AI-driven trade execution in MT4.

### 1.3 Objectives

• Evaluate and compare Java JeroMQ, MQL4 + `libzmq.dll`, and Windows API (`user32.dll`).  
• Analyze key factors such as latency, performance, security, maintainability, and cross-platform support.  
• Justify the final architectural decision based on quantitative and qualitative analysis.

## 5. Conclusion

This paper presented a comparative analysis of three approaches for integrating AI-powered trading into MT4. While Java JeroMQ is ideal for distributed AI models and Windows API offers system-level control, the most effective approach is MQL4 + `libzmq.dll` due to its low latency, native MT4 integration, and efficient data processing.

Future research may explore a hybrid model where AI computation is done in Java JeroMQ, but trade execution remains in MQL4 with `libzmq.dll` for optimal performance.

## 6. References

ZeroMQ Documentation: https://zeromq.org/

MetaTrader 4 MQL4 Documentation: https://www.mql5.com/en/docs

Java JeroMQ Repository: https://github.com/zeromq/jeromq

Reinforcement Learning for Trading Strategies: https://arxiv.org/abs/1903.03104

High-Frequency Trading Using AI: https://ieeexplore.ieee.org/document/9152740

Security Risks in Automated Trading: https://www.sciencedirect.com/science/article/pii/S016792362100002X

Performance Analysis of ZeroMQ vs Traditional Sockets: https://www.usenix.org/conference/nsdi20/presentation/zero-mq

Machine Learning for Financial Markets: https://www.springer.com/gp/book/9783030395759

Distributed Systems and ZeroMQ: https://distributed-systems.net/

AI-Based Trading Execution Models: https://www.nature.com/articles/s41598-021-82106-5