**TradingView to MT4/MT5 Bridge**

**Overview**

This project creates a secure, low-latency bridge that auto-executes TradingView alerts on MT4/MT5. It supports any instrument, demo/live accounts, and multiple brokers.

The solution consists of two main components:

* **Node.js server**: Listens for TradingView webhook alerts via secure WebSocket (WSS).
* **MT5 Expert Advisor (EA)**: Connects to the server to receive alerts and execute trades automatically.

**Project Structure**

pgsql

CopyEdit

bridge/

├── server/

│ ├── server.js

│ ├── package.json

│ ├── Dockerfile

│ ├── .env

│ └── certs/

│ ├── server.key # SSL private key

│ └── server.crt # SSL certificate

├── ea/

│ └── BridgeEA.mq5 # MT5 Expert Advisor

├── sample/

│ ├── sample\_alert.json

│ └── sign.js

└── README.md

**Step-by-Step Setup on Windows 11 (Localhost)**

**1. Install Required Software**

* **Node.js and npm**  
  Download and install from <https://nodejs.org/>
* **MetaTrader 5**  
  Download from https://www.metatrader5.com/en/download
* **OpenSSL** (for certificate generation)  
  Recommended: Use [Git Bash](https://git-scm.com/download/win) which includes OpenSSL, or install OpenSSL separately:  
  https://slproweb.com/products/Win32OpenSSL.html

**2. Setup Node.js Server**

Open **PowerShell** or **Git Bash** and run:

bash

CopyEdit

cd path\to\bridge\server

npm install

**3. Generate Self-Signed SSL Certificates (Localhost Testing)**

Run the following commands in PowerShell or Git Bash:

bash

CopyEdit

mkdir certs

openssl req -x509 -newkey rsa:4096 -nodes -keyout certs/server.key -out certs/server.crt -days 365 -subj "/CN=localhost"

openssl req -x509 -newkey rsa:4096 -nodes -keyout certs/server.key -out certs/server.crt -days 365 -subj "/CN=localhost/O=MyOrg/OU=MyUnit/L=MyCity/ST=MyState/C=US"

This creates:

* server.key: Private key (keep secret)
* server.crt: Public certificate

Place both files inside bridge/server/certs/.

**4. Configure Environment Variables**

Create a .env file inside bridge/server/ with:

ini

CopyEdit

PORT=3000

SECRET=SuperSecret123

AES\_PASS=MyAESPassphrase

Adjust as needed.

**5. Run the Server**

bash

CopyEdit

node server.js

Server listens on https://localhost:3000 (WSS enabled).

**6. Setup MetaTrader 5 EA**

* Copy BridgeEA.mq5 into your MetaTrader 5 Experts folder.
* Open MetaEditor, compile BridgeEA.mq5.
* Attach the EA to any chart.
* Configure EA inputs:
  + SocketServer: "127.0.0.1"
  + SocketPort: 3000
  + AES\_PASS: Same as .env AES\_PASS
  + RetrySec: 10
* Run the EA. It connects to the server and listens for alerts.

**7. Testing Alerts**

Use the sample JSON (sample/sample\_alert.json) or your own TradingView alerts. Send POST requests to:

bash

CopyEdit

https://localhost:3000/webhook

with encrypted JSON payloads.

**MQL5 Libraries Required**

Make sure you have the following MQL5 libraries installed in MetaEditor before compiling the EA:

| **Library** | **Purpose** | **Notes** |
| --- | --- | --- |
| Socket.mqh | WebSocket or TCP client | Community or custom socket implementation |
| Crypto.mqh | AES-256 CBC encryption/decryption | Supports secure payload handling |
| Base64.mqh | Base64 encode/decode | Often built-in or community available |
| JSON.mqh | JSON parsing and manipulation | Community parsers available on MQL5 forums |

You can find many open-source implementations here:  
https://www.mql5.com/en/code/

**Useful Links**

* Node.js & npm: <https://nodejs.org/>
* MetaTrader 5: https://www.metatrader5.com/en/download
* OpenSSL for Windows: https://slproweb.com/products/Win32OpenSSL.html
* MQL5 Codebase: https://www.mql5.com/en/code
* JSON parser example: https://www.mql5.com/en/code/19554

**Notes**

* For **production**, use valid CA-signed SSL certs instead of self-signed.
* Keep your AES passphrase and webhook secret confidential.
* Logs are saved locally in BridgeHistory.log.
* Customize and extend EA to handle multiple brokers/accounts.

**1. Sample .env File**

This file contains environment variables that configure your server. Here's an example of a .env file:

ini

CopyEdit

# Server Configuration

PORT=3000

SECRET=SuperSecret123

AES\_PASS=MyAESPassphrase

# SSL/TLS Configuration

SSL\_CERT\_PATH=./server/certs/server.crt

SSL\_KEY\_PATH=./server/certs/server.key

**Explanation:**

* **PORT**: The port your server listens on (default: 3000).
* **SECRET**: The secret key used for verifying payloads from TradingView alerts.
* **AES\_PASS**: The passphrase for AES-256 encryption/decryption (must match the EA).
* **SSL\_CERT\_PATH and SSL\_KEY\_PATH**: Paths to your SSL certificate and private key for HTTPS/WSS connections (should match your certs folder).

**2. Sample TradingView Alert Template**

To send TradingView alerts to your server, you will need a webhook with a **JSON payload** that matches what your server expects.

**Example TradingView Alert Template (in JSON format):**

This is an example of how to format your alert in TradingView:

json

CopyEdit

{

"secret": "SuperSecret123",

"cmd": "BUY",

"symbol": "EURUSD",

"lot": 0.1,

"sl": 1.20,

"tp": 1.25,

"magic": 123456,

"trail": 0.03

}

**Explanation:**

* **secret**: Must match the SECRET value in the .env file to validate the alert.
* **cmd**: The action to take (either BUY, SELL, CLOSE, PARTIAL\_CLOSE, or TRAIL).
* **symbol**: The trading symbol (e.g., EURUSD, BTCUSD).
* **lot**: The size of the trade (in lots).
* **sl**: Stop Loss (optional, set to 0 if not needed).
* **tp**: Take Profit (optional, set to 0 if not needed).
* **magic**: Unique identifier for the trade (can be any integer).
* **trail**: The trailing stop in pips (optional, used for TRAIL command).

**How to Use in TradingView:**

1. **Go to TradingView** and create or edit an existing alert.
2. In the **Alert Actions** section, select **Webhook URL** and enter:

bash

CopyEdit

https://localhost:3000/webhook

Replace localhost with the public IP/domain if using in production.

1. Paste the **JSON Payload** into the **Message** field (same as above).

**Final Notes**

* You can **test locally** by setting localhost:3000 in your TradingView alert.
* For **production**, change the URL to your actual server's address.
* Make sure that the secret key in both your .env and TradingView alert matches.

**Example Webhook Request (from TradingView):**

When TradingView sends an alert, it will look like this:

bash

CopyEdit

POST https://localhost:3000/webhook

Content-Type: application/json

{

"secret": "SuperSecret123",

"cmd": "BUY",

"symbol": "EURUSD",

"lot": 0.1,

"sl": 1.20,

"tp": 1.25,

"magic": 123456,

"trail": 0.03

}

**🔹 Option 2: Batch Unblock via PowerShell (Recommended for Many Files)**

If you downloaded many .mqh files (e.g., from GitHub):

1. Open **PowerShell** as Administrator.
2. Navigate to your Include directory:

powershell

CopyEdit

cd "C:\Users\<your\_username>\AppData\Roaming\MetaQuotes\Terminal\<YourTerminalID>\MQL5\Include"

1. Run the unblock command:

powershell

CopyEdit

Get-ChildItem -Recurse \*.mqh | Unblock-File

This will unblock **all .mqh files** in that folder and its subfolders.

**💡 After Unblocking**

1. Restart **MetaEditor** if it's open.
2. Recompile your .mq5 or .mqh script.

Steps to run email\_to\_socket.py

Open your terminal/command prompt.

Navigate to the bridge/ folder where your script is located:

bash

Copy

Edit

cd /path/to/bridge

Example for your Windows path:

bash

Copy

Edit

cd D:\Automated Trading Bridge for TradingView Alerts to MT45\bridge

Make sure Python is installed and accessible:

Check your Python version:

bash

Copy

Edit

python --version

or

bash

Copy

Edit

python3 --version

Install required Python packages (if not already installed):

You need pycryptodome for AES encryption:

bash

Copy

Edit

pip install pycryptodome

Also, make sure your imaplib and email packages are available (these come with Python by default).

Run the script:

bash

Copy

Edit

python email\_to\_socket.py

or

bash

Copy

Edit

python3 email\_to\_socket.py

What this will do:

The script will connect to your Gmail inbox every 10 seconds.

It will read any new unread TradingView alert emails.

Decrypt and send them to your MT5 socket server running on 127.0.0.1:3000.

**What to do in TradingView Alerts for email\_to\_socket.py**

**1. Create Alert on TradingView**

* Open your chart on TradingView.
* Click the **"Alert"** button (usually a clock icon or Alt + A).
* Set your alert conditions (e.g., when price crosses a level, indicator signal, etc).

**2. Alert Actions**

Since you **can’t enable webhook on the Basic Plan**, you will use the **Email notification** option.

* In the alert creation popup, make sure **"Send Email"** is checked.
* Enter your dedicated **email address** in your TradingView alert email settings (e.g., bedisarabpreet@gmail.com or the one you are using in your script).

**Note:** You only need to enter the email once in your TradingView account settings for alert notifications to be sent there. TradingView will send alert emails to your email inbox.

**3. Customize Alert Message (optional but recommended)**

In the **"Message"** field of the alert, you can write a JSON payload similar to what your email\_to\_socket.py expects.

For example, a JSON message might look like:

json

CopyEdit

{

"secret": "SuperSecret123",

"cmd": "BUY",

"symbol": "EURUSD",

"lot": 0.1,

"sl": 1.2000,

"tp": 1.2100,

"magic": 123456

}

* This exact JSON message will be sent in the **body of the email**.
* Your email\_to\_socket.py will parse this JSON from the email body and send it to MT5.

**4. How your email\_to\_socket.py script works with this**

* Your script logs into your Gmail inbox.
* It reads **new unseen emails** (the alerts).
* Extracts the email body (which contains your JSON message).
* Encrypts and sends it to the MT5 socket bridge.
* MT5 Expert Advisor reads it and executes orders.

**1. Make sure your email\_to\_socket.py script is running**

* This script connects to your Gmail inbox, reads new alert emails, decrypts and sends commands to MT5 via socket.
* Run it continuously:

bash

CopyEdit

python3 email\_to\_socket.py

You should see output like:

css

CopyEdit

New Alert: { "secret":"SuperSecret123", "cmd":"BUY", ... }

Sent to MT5.

If you see errors here, fix them first.

**2. Make sure your MT5 Bridge EA is running**

* Open MT5.
* Attach the **BridgeEA.mq5 Expert Advisor** to a chart.
* Enable **AutoTrading** in MT5.
* Check the **Experts** tab and **Journal** tab for messages like:

css

CopyEdit

Socket: OK Time: xx:xx:xx

Decoded JSON: {"secret":"SuperSecret123", "cmd":"BUY", ...}

This confirms that MT5 is connected and received data.

**3. Check if trades are executed**

* Based on the alert command (BUY or SELL), your EA should place orders.
* Check the **Trade** tab or **Positions** tab in MT5 to see if new trades are opened matching the alerts.
* You can also check **History** tab for filled orders.

**Step 3: TradingView Alert (Actual Example)**

Inside TradingView's alert box:

json

CopyEdit

{

"cmd": "BUY",

"symbol": "EURUSD",

"lot": 0.1,

"sl": 20,

"tp": 40

}

You will need **something like email\_to\_socket.py** or **TradingView webhook → WebSocket** bridge to forward the alerts to ws://localhost:8080.