Polymarket MCP & CLI

Handbuch für den MCP Server und das CLI Tool

Version 1.2 – Februar 2025

Dr.Jo's dive in:: by Granaria Publishers

Inhaltsverzeichnis

1 Übersicht

Das Polymarket MCP & CLI Toolkit bietet Zugriff auf Polymarket Prediction Markets über zwei Schnittstellen:

- **MCP Server** – Integration mit Claude Desktop, Continue.dev und anderen MCP-Clients
- **CLI Tool** – Terminal-basierter Zugriff für schnelle Abfragen

1.1 Verwendete APIs

API	Base URL	Funktion
Gamma API	gamma-api.polymarket.com	Market Discovery, Events
CLOB API	clob.polymarket.com	Orderbook, Preise
Data API	data-api.polymarket.com	Positionen, Trades

2 Installation

2.1 Voraussetzungen

```
# Python 3.10+
python3 --version

# Dependencies installieren
pip install httpx aiohttp mcp
```

2.2 MCP Server registrieren

2.2.1 Claude Desktop

Konfiguration in `~/Library/Application Support/Claude/claude_desktop_config.json`:

```
{
  "mcpServers": {
    "polymarket": {
      "command": "python3",
      "args": ["/Users/mac/polymarket_mcp/polymarket_mcp.py"]
    }
  }
}
```

2.2.2 Continue.dev (PyCharm)

Konfiguration in `~/.continue/config.json`:

```
{
  "experimental": {
    "modelContextProtocolServers": [
      {
        "name": "polymarket",
        "command": "python3",
        "args": ["/Users/mac/polymarket_mcp/polymarket_mcp.py"]
      }
```

```
    ]
  }
}
```

3 MCP Server

3.1 Starten

```
python3 /Users/mac/polymarket_mcp/polymarket_mcp.py
```

Der Server kommuniziert über stdio mit dem MCP-Client.

3.2 Verfügbare Tools

3.2.1 Market Discovery

Tool	Beschreibung
`polymarket_list_markets`	Markets auflisten
`polymarket_list_events`	Events auflisten
`polymarket_get_market`	Einzelnen Market abrufen
`polymarket_search_markets`	Markets durchsuchen

3.2.2 Orderbook & Preise

Tool	Beschreibung
`polymarket_get_orderbook`	Orderbook (Bids/Asks)
`polymarket_get_price`	Aktueller Preis
`polymarket_get_midpoint`	Midpoint-Preis

3.2.3 Portfolio & Trading

Tool	Beschreibung
`polymarket_get_positions`	Positionen einer Wallet
`polymarket_get_trades`	Trade History
`polymarket_get_activity`	On-Chain Activity
`polymarket_get_holders`	Top Holders eines Markets

3.2.4 BTC 15m Assistant

3.3 Tool-Parameter Beispiele

```
# Markets auflisten
polymarket_list_markets(
    limit=25,              # Max 100
    offset=0,              # Pagination
    closed=False,          # Geschlossene einbeziehen
```

Tool	Beschreibung
polymarket_btc15m_start	Assistant starten
polymarket_btc15m_stop	Assistant stoppen
polymarket_btc15m_status	Status abrufen
polymarket_btc15m_snapshot	Einmaliger Snapshot

```
    response_format="markdown"
)

# Markets suchen
polymarket_search_markets(
    query="bitcoin",        # Suchbegriff
    limit=25
)

# Positionen abrufen
polymarket_get_positions(
    user_address="0x123...",
    sort_by="CASHPNL",
    sort_direction="DESC"
)
```

4 CLI Tool

4.1 Syntax

```
python polymarket_cli.py <command> [options]
```

4.2 Befehle

4.2.1 Markets auflisten

```
# Top 25 Markets
python polymarket_cli.py markets

# Mit Limit
python polymarket_cli.py markets --limit 10

# Mit Suche
python polymarket_cli.py markets --search bitcoin
```

4.2.2 Einzelnen Market anzeigen

```
python polymarket_cli.py market will-bitcoin-reach-150k
```

4.2.3 Preis und Orderbook

```
# Preis abfragen
python polymarket_cli.py price <TOKEN_ID>
```

```
# Orderbook anzeigen
python polymarket_cli.py orderbook <TOKEN_ID> --depth 5
```

4.2.4 Portfolio

```
# Positionen
python polymarket_cli.py positions 0x123...

# Trades
python polymarket_cli.py trades 0x123... --limit 20
```

4.2.5 Bitcoin Markets

```
python polymarket_cli.py btc
```

5 BTC 15m Trading Assistant

Der BTC 15m Assistant analysiert Bitcoin-Preisbewegungen und Polymarket's "BTC Up/Down 15m" Markets.

5.1 Technische Indikatoren

Indikator	Beschreibung	Bullish	Bearish
RSI	Relative Strength (14)	> 55, steigend	< 45, fallend
MACD	Histogramm	Grün expandiert	Rot expandiert
VWAP	Volume Weighted Avg	Preis $>$ VWAP	Preis $<$ VWAP
Heiken Ashi	Geglättete Kerzen	2+ grün	2+ rot

5.2 Trading-Phasen

Phase	Zeit verbleibend	Edge Threshold	Min. Probability
EARLY	> 10 min	5%	55%
MID	5–10 min	10%	60%
LATE	< 5 min	20%	65%

5.3 Verwendung

```
# Assistant starten
polymarket_btc15m_start()

# Status pruefen
polymarket_btc15m_status()

# Einmaliger Snapshot
polymarket_btc15m_snapshot()
```

```
# Assistant stoppen
polymarket_btc15m_stop()
```

5.4 Decision Engine

Die Trading-Entscheidung basiert auf:

1. **Edge berechnen**: Model Probability − Market Price
2. **Threshold prüfen**: Je nach Phase unterschiedlich
3. **Probability validieren**: Mindestwahrscheinlichkeit erfüllt?
4. **Signal generieren**: ENTER oder NO_TRADE

6 Konfiguration

6.1 MCP Server

```
# polymarket_mcp.py
GAMMA_API_BASE = "https://gamma-api.polymarket.com"
CLOB_API_BASE = "https://clob.polymarket.com"
DATA_API_BASE = "https://data-api.polymarket.com"

DEFAULT_TIMEOUT = 30.0
DEFAULT_LIMIT = 25
MAX_LIMIT = 100
```

6.2 BTC 15m Assistant

```
# btc15m_assistant.py
CONFIG = {
    "symbol": "BTCUSDT",
    "candle_window_minutes": 15,
    "rsi_period": 14,
    "macd_fast": 12,
    "macd_slow": 26,
    "macd_signal": 9,
}
```

7 Troubleshooting

7.1 API Fehler

Code	Ursache	Lösung
404	Resource nicht gefunden	Slug/ID prüfen
429	Rate Limit	Warten, weniger Requests
403	Geo-Restriction	VPN verwenden
Timeout	Langsame Verbindung	Retry

7.2 MCP Server Probleme

```
# Manuell testen
python3 /Users/mac/polymarket_mcp/polymarket_mcp.py

# Dependencies pruefen
pip install httpx aiohttp mcp

# Logs checken (Continue.dev)
cat ~/.continue/logs/core.log | tail -50
```

8 Erweiterte Nutzung

8.1 Kombinierte Workflows

8.1.1 Market Research Pipeline

Vollständiger Research-Workflow:

```
import asyncio
import httpx

class PolymarketResearch:
    def __init__(self):
        self.gamma = "https://gamma-api.polymarket.com"
        self.clob = "https://clob.polymarket.com"
        self.data = "https://data-api.polymarket.com"

    async def research_market(self, search_query: str):
        async with httpx.AsyncClient(timeout=30) as client:
            # 1. Market suchen
            markets = await client.get(
                f"{self.gamma}/markets",
                params={"_q": search_query, "_limit": 5}
            )
            market = markets.json()[0]

            # 2. Orderbook analysieren
            yes_token = next(
                (t for t in market.get("tokens", [])
                 if t.get("outcome") == "Yes"), None
            )

            if yes_token:
                book = await client.get(
                    f"{self.clob}/book",
                    params={"token_id": yes_token["token_id"]}
                )
                return {
                    "market": market["question"],
                    "orderbook": book.json()
                }

asyncio.run(PolymarketResearch().research_market("bitcoin"))
```

8.1.2 Portfolio Monitoring

```python
class PortfolioMonitor:
    def __init__(self, wallet: str, threshold: float = 0.10):
        self.wallet = wallet
        self.threshold = threshold
        self.last_prices = {}

    async def check_alerts(self):
        # Positionen abrufen
        positions = await self.get_positions()
        alerts = []

        for pos in positions:
            current = await self.get_price(pos["tokenId"])
            if pos["tokenId"] in self.last_prices:
                change = (current - self.last_prices[pos["tokenId"]])
                        / self.last_prices[pos["tokenId"]]
                if abs(change) >= self.threshold:
                    alerts.append(f"{pos['title']}: {change*100:+.1f}%")

            self.last_prices[pos["tokenId"]] = current
        return alerts
```

8.2 Performance-Optimierung

8.2.1 Caching

```python
from datetime import datetime, timedelta

class CachedClient:
    def __init__(self, ttl_seconds=60):
        self.cache = {}
        self.ttl = timedelta(seconds=ttl_seconds)

    async def get_cached(self, url, params=None):
        key = f"{url}:{params}"
        if key in self.cache:
            time, data = self.cache[key]
            if datetime.now() - time < self.ttl:
                return data

        data = await self._fetch(url, params)
        self.cache[key] = (datetime.now(), data)
        return data
```

8.2.2 Batch Requests

```python
class BatchClient:
    def __init__(self, max_concurrent=10):
        self.semaphore = asyncio.Semaphore(max_concurrent)

    async def get_prices_batch(self, token_ids):
        async with httpx.AsyncClient() as client:
            tasks = [self._fetch_price(client, tid)
                     for tid in token_ids]
            return await asyncio.gather(*tasks)
```

9 FAQ

MCP vs CLI? MCP für AI-Integration (Claude, Continue.dev), CLI für Terminal.

Account nötig? Nur zum Traden. Lesen ist ohne Account möglich.

403 Fehler? Geo-Restriction – VPN zu erlaubter Region nutzen.

Was ist Edge? Edge = Modell-Wahrscheinlichkeit − Markt-Preis

NO_TRADE Signal? Edge unter Threshold oder Regime = CHOP

Rate Limits? Keine offiziellen, aber >100/min kann blocken

10 Glossar

Begriff	Erklärung
Condition ID	Eindeutige ID eines Markets auf der Blockchain
CLOB	Central Limit Order Book
Edge	Differenz zwischen Modell und Marktpreis
Gamma API	Polymarket API für Market Discovery
Heiken Ashi	Kerzentyp der Preisbewegungen glättet
Liquidity	Verfügbare Liquidität im Orderbook
MACD	Moving Average Convergence Divergence
MCP	Model Context Protocol
Midpoint	Mitte zwischen bestem Bid und Ask
Outcome	Mögliches Ergebnis (Yes/No)
RSI	Relative Strength Index
Slug	URL-freundlicher Market-Identifier
Token ID	Eindeutige ID eines Outcome-Tokens
VWAP	Volume Weighted Average Price

11 Ressourcen

11.1 Offizielle Dokumentation

- Polymarket Docs: `https://docs.polymarket.com/`
- MCP Specification: `https://modelcontextprotocol.io/`

11.2 APIs

- Gamma: `https://gamma-api.polymarket.com`
- CLOB: `https://clob.polymarket.com`
- Data: `https://data-api.polymarket.com`

12 Sicherheitshinweise

- **API-Keys:** Nie in Code committen, Umgebungsvariablen nutzen
- **Wallet:** Nie Private Keys teilen, dedizierte Trading-Wallet

- **Rate Limiting:** Polling >1s, Caching implementieren
- **Datenschutz:** Wallet-Adressen und Positionen sind öffentlich

13 Dateistruktur

```
/Users/mac/polymarket_mcp/
    polymarket_mcp.py       # MCP Server (809 Zeilen)
    polymarket_cli.py       # CLI Tool (378 Zeilen)
    btc15m_assistant.py     # Trading Assistant (1005 Zeilen)
    requirements.txt        # Dependencies
    pyproject.toml          # Project Config
    docs/
        HANDBUCH.md         # Markdown Version
        HANDBUCH.tex        # LaTeX Version (dieses Dokument)
```

14 Quick Reference

14.1 MCP Tools

```
MARKET DISCOVERY
  polymarket_list_markets(limit, offset, closed)
  polymarket_list_events(limit, offset, closed)
  polymarket_get_market(slug)
  polymarket_search_markets(query, limit)

ORDERBOOK & PRICES
  polymarket_get_orderbook(token_id)
  polymarket_get_price(token_id)
  polymarket_get_midpoint(token_id)

PORTFOLIO & TRADING
  polymarket_get_positions(user_address, sort_by)
  polymarket_get_trades(user_address, limit)
  polymarket_get_activity(user_address, activity_type)
  polymarket_get_holders(market_id, limit)

BTC 15m ASSISTANT
  polymarket_btc15m_start(poll_interval_ms)
  polymarket_btc15m_stop()
  polymarket_btc15m_status()
  polymarket_btc15m_snapshot()
```

14.2 CLI Commands

```
python polymarket_cli.py markets [-l N] [-s QUERY] [--closed]
python polymarket_cli.py market <slug>
python polymarket_cli.py events [-l N] [--closed]
python polymarket_cli.py price <token_id>
python polymarket_cli.py orderbook <token_id> [-d N]
python polymarket_cli.py positions <wallet>
python polymarket_cli.py trades <wallet> [-l N]
python polymarket_cli.py btc
```

14.3 BTC Assistant Signals

Phase	Time Left	Edge Threshold	Min Prob
EARLY	> 10 min	5%	55%
MID	5–10 min	10%	60%
LATE	< 5 min	20%	65%