**Logs.**

Existem duas versões dos programas. Uma com muitos logs para facilitar o desenvolvimento e a depuração e outra com poucos logs, que é para acompanhar a execução, quando os programas já foram depurados. Os logados estão na pasta ...\Tester\VersoesAnterioresOK e terminam em Log, como worker...Log.js. Os outros estão com os nomes normais no \tester.

**Logs Padronizados para o Novo Worker de Execução na Fase Desenvolvimento.**

Os logs devem ser enviados via parentPort.postMessage (para logs gerais e depuração) e, quando aplicável, via portToCoord.postMessage (para status de ordens relevantes ao coordenador). Cada log deve incluir um type para categorização, um timestamp absoluto (abs, em milissegundos desde o Unix epoch), e, quando relevante, um tempo relativo (t, em milissegundos desde t0, inicializado pelo coordenador). Os logs devem ser concisos, mas informativos, com detalhes suficientes para diagnosticar problemas.

1. **Inicialização do Worker**
   * **Descrição**: Confirma que o worker foi inicializado e está pronto para operar.
   * **Formato**:

{ type: 'workerInitialized',

exchange: '<exchange\_name>',

abs: Date.now(),

message: 'Worker <exchange\_name> inicializado' }

* + **Canal**: parentPort
  + **Quando**: Após a configuração inicial (ex.: criação do orderIdMap, inicialização do WebSocket).
  + **Exemplo**: { type: 'workerInitialized', exchange: 'NewExchange', abs: 1734567890123, message: 'Worker NewExchange inicializado' }

1. **Conexão WebSocket Estabelecida**
   * **Descrição**: Indica que o WebSocket foi conectado com sucesso à exchange.
   * **Formato**: {

type: 'wsConnected',

exchange: '<exchange\_name>',

abs: Date.now(),

wsUrl: '<websocket\_url>',

message: 'WebSocket conectado'

}

* + **Canal**: parentPort
  + **Quando**: Após o evento de abertura do WebSocket (ex.: ws.on('open')).
  + **Exemplo**: { type: 'wsConnected', exchange: 'NewExchange', abs: 1734567890123, wsUrl: 'wss://ws.newexchange.com/v1/private', message: 'WebSocket conectado' }

1. **Autenticação WebSocket**
   * **Descrição**: Confirma o sucesso ou falha na autenticação do WebSocket.
   * **Formato** (Sucesso): {

type: 'wsAuthenticated',

exchange: '<exchange\_name>',

abs: Date.now(),

message: 'WebSocket autenticado com sucesso'

}

* + **Formato** (Falha): {

type: 'wsAuthFailed',

exchange: '<exchange\_name>',

abs: Date.now(),

error: '<error\_message>',

message: 'Falha na autenticação do WebSocket'

}

* + **Canal**: parentPort
  + **Quando**: Após a resposta da exchange à mensagem de autenticação (ex.: HMAC-SHA256 com apiKey).
  + **Exemplo** (Sucesso): { type: 'wsAuthenticated', exchange: 'NewExchange', abs: 1734567890123, message: 'WebSocket autenticado com sucesso' }
  + **Exemplo** (Falha): { type: 'wsAuthFailed', exchange: 'NewExchange', abs: 1734567890123, error: 'Invalid API key', message: 'Falha na autenticação do WebSocket' }

1. **Subscrição de Canais**
   * **Descrição**: Registra a subscrição bem-sucedida ou falha nos canais WebSocket (ex.: ordens, status).
   * **Formato** (Sucesso): {

type: 'channelSubscribed',

exchange: '<exchange\_name>',

abs: Date.now(),

channel: '<channel\_name>',

message: 'Canal <channel\_name> subscrito'

}

* + **Formato** (Falha): {

type: 'channelSubscriptionFailed',

exchange: '<exchange\_name>',

abs: Date.now(),

channel: '<channel\_name>',

error: '<error\_message>',

message: 'Falha na subscrição do canal <channel\_name>'

}

* + **Canal**: parentPort
  + **Quando**: Após a confirmação da exchange sobre a subscrição de cada canal.
  + **Exemplo** (Sucesso): { type: 'channelSubscribed', exchange: 'NewExchange', abs: 1734567890123, channel: 'orders', message: 'Canal orders subscrito' }
  + **Exemplo** (Falha): { type: 'channelSubscriptionFailed', exchange: 'NewExchange', abs: 1734567890123, channel: 'orders', error: 'Invalid channel', message: 'Falha na subscrição do canal orders' }

1. **Recebimento de Ordem**
   * **Descrição**: Confirma que o worker recebeu uma ordem do coordenador para execução.
   * **Formato**: {

type: 'orderReceived',

exchange: '<exchange\_name>',

abs: Date.now(),

clientOrderId: '<clientOrderId>',

order: { symbol: '<symbol>', side: '<side>', type: '<type>', amount: <amount>, price: <price> },

message: 'Ordem recebida para execução'

}

* + **Canal**: parentPort
  + **Quando**: Ao receber uma mensagem { type: 'executeOrder' } via portToCoord.
  + **Exemplo**: { type: 'orderReceived', exchange: 'NewExchange', abs: 1734567890123, clientOrderId: 'NEWEX-1734567890123', order: { symbol: 'BTCUSDT', side: 'buy', type: 'market', amount: 0.001, price: null }, message: 'Ordem recebida para execução' }

1. **Postagem de Ordem**
   * **Descrição**: Registra o resultado da postagem de uma ordem (aceita ou rejeitada).
   * **Formato** (Aceita): {

type: 'orderPosted',

exchange: '<exchange\_name>',

abs: Date.now(),

clientOrderId: '<clientOrderId>',

orderId: '<exchange\_orderId>',

instId: '<symbol>',

message: 'Ordem postada com sucesso'

}

* + **Formato** (Rejeitada): {

type: 'orderRejected',

exchange: '<exchange\_name>',

abs: Date.now(),

clientOrderId: '<clientOrderId>',

instId: '<symbol>',

errorCode: '<error\_code>',

errorMessage: '<error\_message>',

message: 'Ordem rejeitada'

}

* + **Canal**: parentPort e portToCoord (para orderPosted e orderRejected, com status accepted ou rejected)
  + **Quando**: Após a resposta da exchange à tentativa de postagem.
  + **Exemplo** (Aceita): { type: 'orderPosted', exchange: 'NewExchange', abs: 1734567890123, clientOrderId: 'NEWEX-1734567890123', orderId: '123456789', instId: 'BTCUSDT', message: 'Ordem postada com sucesso' }
  + **Exemplo** (Rejeitada): { type: 'orderRejected', exchange: 'NewExchange', abs: 1734567890123, clientOrderId: 'NEWEX-1734567890123', instId: 'BTCUSDT', errorCode: '51020', errorMessage: 'Order amount too low', message: 'Ordem rejeitada' }

1. **Atualização de Status da Ordem**
   * **Descrição**: Reporta mudanças no estado da ordem no mercado (ex.: ativa, preenchida, cancelada).
   * **Formato**: {

type: 'orderStatus',

exchange: '<exchange\_name>',

abs: Date.now(),

t: <relative\_time>,

clientOrderId: '<clientOrderId>',

orderId: '<exchange\_orderId>',

instId: '<symbol>',

status: '<live|filled|cancelled>',

avgPx: <average\_price|null>,

fillSz: <filled\_size|null>,

cancelSource: '<cancel\_reason|null>',

message: 'Atualização de status da ordem'

}

* + **Canal**: portToCoord (com cópia resumida para parentPort para depuração)
  + **Quando**: Ao receber atualizações do canal de ordens da exchange (ex.: filled, cancelled).
  + **Exemplo**: { type: 'orderStatus', exchange: 'NewExchange', abs: 1734567890123, t: 150, clientOrderId: 'NEWEX-1734567890123', orderId: '123456789', instId: 'BTCUSDT', status: 'filled', avgPx: 50000, fillSz: 0.001, cancelSource: null, message: 'Atualização de status da ordem' }

1. **Erro de WebSocket**
   * **Descrição**: Registra erros no WebSocket (ex.: desconexão, falha de mensagem).
   * **Formato**: {

type: 'wsError',

exchange: '<exchange\_name>',

abs: Date.now(),

error: '<error\_message>',

message: 'Erro no WebSocket'

}

* + **Canal**: parentPort
  + **Quando**: Ao capturar um erro no WebSocket (ex.: ws.on('error')).
  + **Exemplo**: { type: 'wsError', exchange: 'NewExchange', abs: 1734567890123, error: 'Connection reset by peer', message: 'Erro no WebSocket' }

1. **Reconexão WebSocket**
   * **Descrição**: Indica que o worker está tentando reconectar ao WebSocket após uma falha.
   * **Formato**: {

type: 'wsReconnecting',

exchange: '<exchange\_name>',

abs: Date.now(),

attempt: <attempt\_number>,

message: 'Tentando reconectar WebSocket'

}

* + **Canal**: parentPort
  + **Quando**: Ao iniciar uma tentativa de reconexão (ex.: após 5s de falha).
  + **Exemplo**: { type: 'wsReconnecting', exchange: 'NewExchange', abs: 1734567890123, attempt: 1, message: 'Tentando reconectar WebSocket' }

1. **Ping/Pong WebSocket**
   * **Descrição**: Confirma que o ping foi enviado e/ou pong recebido para manter a conexão ativa.
   * **Formato**: {

type: 'wsPingPong',

exchange: '<exchange\_name>',

abs: Date.now(),

message: 'Ping enviado/Pong recebido'

}

* + **Canal**: parentPort
  + **Quando**: Ao enviar um ping (ex.: a cada 20-25s) ou receber um pong.
  + **Exemplo**: { type: 'wsPingPong', exchange: 'NewExchange', abs: 1734567890123, message: 'Ping enviado/Pong recebido' }

**Notas sobre a Implementação dos Logs dos Workers**

* **Nível de Detalhe**: Os logs devem ser suficientemente detalhados para rastrear o fluxo (ex.: clientOrderId, orderId, instId) e diagnosticar erros (ex.: errorCode, errorMessage), mas evitar sobrecarga com informações redundantes.
* **Canais**:
  + Use parentPort para logs de depuração e eventos gerais (ex.: conexão, erros).
  + Use portToCoord para logs relacionados ao fluxo de ordens (orderPosted, orderRejected, orderStatus) que o coordenador precisa processar.
* **Formato JSON**: Todos os logs devem ser objetos JSON para facilitar parsing e análise automatizada.
* **Campos Obrigatórios**: Todo log deve incluir type, exchange, abs, e message. Outros campos (ex.: t, clientOrderId) são específicos ao contexto.
* **Depuração Inicial**: Durante o desenvolvimento, logs como orderReceived e orderPosted são críticos para verificar a integração com o coordenador simulado ou mainProxy.
* **Logs no Coordenador**: Se problemas difíceis de diagnosticar surgirem (ex.: ordens não processadas corretamente), logs semelhantes podem ser adicionados ao coordenador, focando em eventos como envio de ordens, recebimento de status e timeouts.

**Logs Padronizados para o Coordenador na Fase Desenvolvimento.**

**Descrição Geral :**

Os logs são projetados para facilitar a depuração e rastreamento no Worker Coordenador, cobrindo inicialização, envio de ordens, recebimento de status, gerenciamento de timeouts, erros e decisões de arbitragem. Todos os logs são enviados via parentPort.postMessage para o main ou sistema de logging. Cada log inclui type, abs (timestamp absoluto em ms desde Unix epoch), e, quando relevante, t (tempo relativo em ms desde t0, inicializado ao enviar T1). Os logs são em formato JSON para fácil parsing.

1. Inicialização do Coordenador

Descrição: Confirma que o coordenador foi inicializado e configurou as conexões com os workers.

Formato:

{

type: 'coordinatorInitialized',

abs: Date.now(),

exchanges: ['<exchange1>', '<exchange2>', '...'],

message: 'Coordenador inicializado com workers'

}

Canal: parentPort

Quando: Após criar workers (ex.: OKX, Bybit, Binance) e configurar MessageChannel.

Exemplo: { type: 'coordinatorInitialized', abs: 1734567890123, exchanges: ['OKX', 'Bybit', 'Binance'], message: 'Coordenador inicializado com workers' }

2. Matriz de Operações Recebida/Simulada

Descrição: Registra o recebimento ou simulação da matriz de operações (T1, T2, T3).

Formato:

{

type: 'matrixProcessed',

abs: Date.now(),

t0: <t0\_timestamp>,

matrix: [{ exchange: '<exchange>', symbol: '<symbol>', side: '<side>', type: '<type>', amount: <amount>, price: <price> }, ...],

message: 'Matriz de operações recebida/simulada'

}

Canal: parentPort

Quando: Ao receber a matriz do main ou após simular internamente.

Exemplo: { type: 'matrixProcessed', abs: 1734567890123, t0: 1734567890000, matrix: [{ exchange: 'OKX', symbol: 'BTC-USDT', side: 'buy', type: 'market', amount: 0.001, price: null }, ...], message: 'Matriz de operações recebida/simulada' }

3. Envio de Ordem

Descrição: Confirma o envio de uma ordem (T1, T2 ou T3) a um worker de execução.

Formato:

{

type: 'orderSent',

abs: Date.now(),

t: <relative\_time>,

clientOrderId: '<clientOrderId>',

exchange: '<exchange>',

order: { symbol: '<symbol>', side: '<side>', type: '<type>', amount: <amount>, price: <price> },

orderTag: '<T1|T2|T3>',

message: 'Ordem enviada ao worker'

}

Canal: parentPort

Quando: Ao enviar { type: 'executeOrder' } via portToCoord para um worker.

Exemplo: { type: 'orderSent', abs: 1734567890123, t: 0, clientOrderId: 'COORD-1734567890123', exchange: 'OKX', order: { symbol: 'BTC-USDT', side: 'buy', type: 'market', amount: 0.001, price: null }, orderTag: 'T1', message: 'Ordem enviada ao worker' }

4. Recebimento de Status de Ordem

Descrição: Registra o status de uma ordem recebido de um worker (accepted, rejected, filled, cancelled).

Formato:

{

type: 'orderStatusReceived',

abs: Date.now(),

t: <relative\_time>,

clientOrderId: '<clientOrderId>',

exchange: '<exchange>',

status: '<accepted|rejected|filled|cancelled>',

instId: '<symbol>',

avgPx: <average\_price|null>,

fillSz: <filled\_size|null>,

cancelSource: '<cancel\_reason|null>,

orderTag: '<T1|T2|T3>',

message: 'Status de ordem recebido'

}

Canal: parentPort

Quando: Ao receber { type: 'orderStatus' } via portToCoord de um worker.

Exemplo: { type: 'orderStatusReceived', abs: 1734567890123, t: 150, clientOrderId: 'COORD-1734567890123', exchange: 'OKX', status: 'filled', instId: 'BTC-USDT', avgPx: 50000, fillSz: 0.001, cancelSource: null, orderTag: 'T1', message: 'Status de ordem recebido' }

5. Decisão de Arbitragem

Descrição: Registra a decisão tomada com base no status de T1 (prosseguir com T2/T3 ou abortar).

Formato (Prosseguir):

{

type: 'arbitrageDecision',

abs: Date.now(),

t: <relative\_time>,

clientOrderId: '<clientOrderId>',

orderTag: 'T1',

decision: 'proceed',

message: 'T1 preenchida, prosseguindo com T2 e T3'

}

Formato (Abortar):

{

type: 'arbitrageDecision',

abs: Date.now(),

t: <relative\_time>,

clientOrderId: '<clientOrderId>',

orderTag: 'T1',

decision: 'abort',

reason: '<rejected|cancelled|timeout>',

message: 'Arbitragem abortada'

}

Canal: parentPort

Quando: Após processar o status de T1 e decidir o fluxo (prosseguir ou abortar).

Exemplo (Prosseguir): { type: 'arbitrageDecision', abs: 1734567890123, t: 150, clientOrderId: 'COORD-1734567890123', orderTag: 'T1', decision: 'proceed', message: 'T1 preenchida, prosseguindo com T2 e T3' }

Exemplo (Abortar): { type: 'arbitrageDecision', abs: 1734567890123, t: 150, clientOrderId: 'COORD-1734567890123', orderTag: 'T1', decision: 'abort', reason: 'rejected', message: 'Arbitragem abortada' }

6. Timeout de Ordem

Descrição: Indica que o timeout de 3s para T1 foi atingido, resultando em aborto.

Formato:

{

type: 'orderTimeout',

abs: Date.now(),

t: <relative\_time>,

clientOrderId: '<clientOrderId>',

exchange: '<exchange>',

orderTag: 'T1',

message: 'Timeout de 3s atingido, arbitragem abortada'

}

Canal: parentPort

Quando: Ao detectar que T1 não foi preenchida dentro de 3s.

Exemplo: { type: 'orderTimeout', abs: 1734567890123, t: 3000, clientOrderId: 'COORD-1734567890123', exchange: 'OKX', orderTag: 'T1', message: 'Timeout de 3s atingido, arbitragem abortada' }

7. Erro de Comunicação com Worker

Descrição: Registra falhas na comunicação com um worker de execução (ex.: MessageChannel fechado).

Formato:

{

type: 'workerCommError',

abs: Date.now(),

exchange: '<exchange>',

error: '<error\_message>',

message: 'Erro na comunicação com worker'

}

Canal: parentPort

Quando: Ao capturar um erro ao enviar ou receber mensagens via portToCoord.

Exemplo: { type: 'workerCommError', abs: 1734567890123, exchange: 'OKX', error: 'MessageChannel closed', message: 'Erro na comunicação com worker' }

8. Finalização da Arbitragem

Descrição: Confirma a conclusão de um ciclo de arbitragem (bem-sucedido ou abortado).

Formato (Bem-sucedido):

{

type: 'arbitrageCompleted',

abs: Date.now(),

t: <relative\_time>,

clientOrderIds: ['<clientOrderId\_T1>', '<clientOrderId\_T2>', '<clientOrderId\_T3>'],

message: 'Ciclo de arbitragem concluído com sucesso'

}

Formato (Abortado):

{

type: 'arbitrageAborted',

abs: Date.now(),

t: <relative\_time>,

clientOrderId: '<clientOrderId>',

reason: '<rejected|cancelled|timeout>',

message: 'Ciclo de arbitragem abortado'

}

Canal: parentPort

Quando: Após completar T1, T2, T3 (bem-sucedido) ou abortar o ciclo.

Exemplo (Bem-sucedido): { type: 'arbitrageCompleted', abs: 1734567890123, t: 450, clientOrderIds: ['COORD-1734567890123', 'COORD-1734567890140', 'COORD-1734567890150'], message: 'Ciclo de arbitragem concluído com sucesso' }

Exemplo (Abortado): { type: 'arbitrageAborted', abs: 1734567890123, t: 150, clientOrderId: 'COORD-1734567890123', reason: 'timeout', message: 'Ciclo de arbitragem abortado' }

**Notas sobre a Implementação dos Logs do Coordenador:**

- Nível de Detalhe: Os logs devem rastrear o fluxo de ordens (via clientOrderId e orderTag), decisões de arbitragem e erros, com informações suficientes para identificar falhas (ex.: motivo de aborto, erros de comunicação).

- Canal: Todos os logs são enviados via parentPort, pois o coordenador se comunica com o main e não precisa enviar logs aos workers. Logs de status de ordens são recebidos dos workers via portToCoord, mas registrados via parentPort para depuração.

- Formato JSON: Todos os logs são objetos JSON para facilitar parsing e análise.

- Campos Obrigatórios: Todo log inclui type, abs, e message. Campos como t, clientOrderId, exchange, e orderTag são contextuais.

- Depuração: Logs como orderSent, orderStatusReceived, e arbitrageDecision são essenciais para verificar a integração com workers e o fluxo de arbitragem.

- Uso no Desenvolvimento: Esses logs ajudam a diagnosticar problemas como ordens não enviadas, status não recebidos, ou timeouts, especialmente ao integrar novos workers ou simular matrizes.

**Logs Padronizados para Coordenador e Workers Operacionais.**

Uma vez que os programas já foram desenvolvidos/debugados e se encontram operacionais os logs são apenas informativos para acompanhar a execução. Neste caso, a quantidade de logs é minimizada e responde aos seguintes critérios.

Logs dos Workers:

Mensagem de Inicializacao do worker.

Confirmacao da conexao (conexao pronta)

Mensagem de postagem da ordem enviado à plataforma identificando o T(1,2 ou3)

Resposta completa da plataforma para a VALIDACAO da ordem (na Binance esta mensagem nao existe).

Resposta completa da plataforma para a EXECUCAO da ordem.

Logs do Coordenador:

Mensagem de todas as conexoes prontas

Matriz de operacoes (tempo = 0)

Envio de T1 (nao precisa printar detalhes da ordem porque eles sao printados pelo worker)

Resposta de T1 se a arbitragem continua ou é abortada. Avisar se foi pelo timeout (se foi rejected ou cancelled ja avisou o worker).

Resultado final das 3 operacoes (abortada, nao enviada, rejected, filled/cancelled/live, etc.)