**Estrutura da Binance para WebSockets**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Pelo que entendemos ate agora, nao existe um canal de ws para postagem de ordens com margem. A única  \* forma de fazer isso é atravez de REST. A informacao veio por uma resposta no forum dos desenvolvedores. Para determinar uma operacao em margem a conta de margin deve estar configurada e ter fundos. A ord em no REST deve conter estes dois campos:**

**isIsolated: 'FALSE', (issolada/cruzada)**

**sideEffectType: 'AUTO\_BORROW', (tem mais opções)**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

A Binance oferece dois tipos principais de WebSockets, cada um com propósitos distintos:

1. **WebSocket Streams (Data Streams):**
   * **Endpoint:** wss://stream.binance.com:9443/ws/<streamName> (ou wss://stream.binance.com:443 para alguns casos).
   * **Propósito:** Fornece dados em tempo real, como preços de mercado, profundidade do livro de ordens, trades ou eventos autenticados da conta do usuário (User Data Stream).
   * **Autenticação:** Para streams públicos (ex.: preços), não precisa de chave API. Para streams privados (ex.: dados da conta), você precisa de uma chave API e um "listen key".
   * **Exemplo de uso:** Escutar atualizações da conta (saldo, ordens, posições).
2. **WebSocket API (Trading API):**
   * **Endpoint:** wss://ws-api.binance.com:443/ws-api/v3.
   * **Propósito:** Permite interação ativa com a plataforma, como criar ordens, cancelar ordens, consultar saldo, etc. É uma alternativa mais rápida à REST API.
   * **Autenticação:** Requer chave API e assinatura HMAC-SHA256 para cada comando.
   * **Exemplo de uso:** Enviar uma ordem de compra ou venda.

Para o seu caso (enviar uma ordem e receber respostas), precisamos usar **ambos os WebSockets**, porque:

* O **WebSocket API** é usado para *enviar* a ordem (order.place).
* O **User Data Stream** é usado para *escutar* os eventos resultantes (como executionReport para status da ordem e outboundAccountPosition para posições da conta).

**WebSockets Necessários**

1. **WebSocket API (wss://ws-api.binance.com:443/ws-api/v3):**
   * **Função:** Enviar a ordem de compra limite (ETHUSDT, BUY, LIMIT, 0.00540000, 1281.46000000).
   * **Método:** order.place.
   * **Resposta Imediata:** Após enviar a ordem, o WebSocket API retorna uma confirmação (ex.: NEW se aceita, ou erro como REJECTED se rejeitada).
   * **Autenticação:** Usa apiKey e signature (calculada com apiSecret).
2. **User Data Stream (wss://stream.binance.com:9443/ws/<listenKey>):**
   * **Função:** Escutar eventos em tempo real da conta após a ordem ser enviada.
   * **Canal Específico:** Para Cross Margin, usamos o método subscribeCrossMarginUserDataStream() para obter o listenKey e conectar ao stream autenticado.
   * **Autenticação:** Requer um listenKey, que é gerado via REST API (POST /sapi/v1/userDataStream/isolated ou similar) ou pela biblioteca.

**Eventos e Canais Necessários**

Aqui estão os eventos e canais que você precisa para enviar a ordem e receber as respostas:

**1. Envio da Ordem (WebSocket API)**

* **Canal:** Não há "canal" específico no WebSocket API; você envia um comando JSON como:

{

"id": "12345",

"method": "order.place",

"params": {

"symbol": "ETHUSDT",

"side": "BUY",

"type": "LIMIT",

"quantity": "0.00540000",

"price": "1281.46000000",

"timeInForce": "GTC",

"timestamp": <timestamp>,

"apiKey": "<sua-chave>",

"signature": "<assinatura>"

}

}

* **Resposta Esperada:** Um objeto JSON com o status inicial da ordem, como:

{

"id": "12345",

"status": 200,

"result": {

"symbol": "ETHUSDT",

"orderId": 27959806866,

"clientOrderId": "web\_98f97602960d44b8bb0cb7b508904aa1",

"status": "NEW",

"side": "BUY",

"type": "LIMIT",

"quantity": "0.00540000",

"price": "1281.46000000"

}

}

* + Se rejeitada, retorna um erro como "code": -2010, "msg": "Account has insufficient balance".

**2. Escuta de Eventos (User Data Stream)**

* **Canal:** crossMargin\_userData\_<listenKey> (gerado por subscribeCrossMarginUserDataStream()).
* **Eventos Relevantes:**
  1. **executionReport:**
     + **Descrição:** Informa o status da ordem em tempo real.
     + **Status Possíveis (campos x e X):**
       - NEW: Ordem criada.
       - FILLED: Ordem totalmente executada.
       - PARTIALLY\_FILLED: Ordem parcialmente executada.
       - CANCELED: Ordem cancelada.
       - REJECT nationalistED: Ordem rejeitada (geralmente acompanhada de um erro).
     + **Exemplo (baseado na sua saída):**

{

"e": "executionReport",

"s": "ETHUSDT",

"S": "BUY",

"o": "LIMIT",

"f": "GTC",

"q": "0.00540000",

"p": "1281.46000000",

"x": "NEW",

"X": "NEW",

...

}

* 1. **outboundAccountPosition:**
     + **Descrição:** Atualiza as posições/saldos da conta após a ordem afetar o saldo (ex.: borrow ou execução).
     + **Exemplo (baseado na sua saída):**

{

"e": "outboundAccountPosition",

"B": [

{ "a": "ETH", "f": "0.00000028", "l": "0.00000000" },

{ "a": "USDT", "f": "10.69268224", "l": "0.00000000" }

],

...

}

* + - Aqui, f é o saldo livre e l é o saldo bloqueado (locked).

**O que Você Precisa no Código**

Para enviar a ordem e receber todas as respostas que mencionou (rejected, new, filled, canceled, e o array de posições):

1. **Conectar ao WebSocket API (wss://ws-api.binance.com:443/ws-api/v3):**
   * Enviar o comando order.place com os parâmetros da ordem.
   * Escutar a resposta inicial (confirmação ou rejeição).
2. **Conectar ao User Data Stream (wss://stream.binance.com:9443/ws/<listenKey>):**
   * Usar subscribeCrossMarginUserDataStream() para escutar eventos.
   * Filtrar e exibir:
     + executionReport com os campos s, S, o, f, q, p, x, X.
     + outboundAccountPosition para mostrar as posições da conta (array B).
3. **Considerações Adicionais:**
   * **Auto-Borrow:** Como discutimos, isso é automático se o saldo for insuficiente e a conta estiver configurada para Cross Margin com "auto-borrow" ativado. Não precisamos especificar no order.place.
   * **Listen Key:** O subscribeCrossMarginUserDataStream() gera o listenKey automaticamente na biblioteca binance, mas ele expira em 60 minutos se não for mantido vivo (via ping ou REST).

**Resumo dos WebSockets e Eventos**

* **WebSockets:**
  1. wss://ws-api.binance.com:443/ws-api/v3 (enviar ordem).
  2. wss://stream.binance.com:9443/ws/<listenKey> (escutar eventos).
* **Eventos:**
  1. order.place resposta (via WebSocket API): Confirmação inicial ou rejeição.
  2. executionReport (via User Data Stream): Status da ordem (NEW, FILLED, CANCELED, etc.).
  3. outboundAccountPosition (via User Data Stream): Posições da conta após a ordem.