Coindeal API-Documentation

Table of Contents

Offer-table

This micro-service is most important for trading functionality API to client application.
It sends informations about aggregated order and transactions lists for market (market is pair of currencies for example **BTC/USD**) and brodcasts to connected clients informations about new orders and transactions. Also when user want create an offer needs to use this service.

How to listen over WebSocket (SocketCluster)?

1. Download and implement to your app socket liblary client SocketCluster.

2. Using this client start subscribing channel

    - `aggregated-offer` - incoming offers
    - `init-aggregated-offers` - initial aggregated offers to market (sorted by unit price)
    - `transaction` - incoming transactions (not aggregated)
    - `transactions` - aggregated transactions list
    - `exchange-state@{base-currency}-{quote-currency}` - state of market for pair, for example: `exchange-state@LTC-USD`

- `theoretical-opening-price@{base-currency}-{quote-currency}` - theoretical opening price for pair, for example: `theoretical-opening-price@LTC-USD`

3. Watch those channels for react for events

Orders

For getting full state of orders you are required to fetch current state of market (historical orders) - for that you need subsribe `init-aggregated-offers` channel.

Subscription to `aggregated-offer` channel makes that we will recive only orders which parameters have changed since starting subscription.

For better understanding, I will add sample snipfets in `javascript`

```
var socket = socketCluster.connect();

socket.emit('init-aggregated-offers', stockExchange, funct
ion(errCode, response) {
    if (errCode) {
        // error occurred - details are in response
    } else {
        // success
        response.forEach(function(stockMarketAggregatedOff
ersDto) {
            // handle StockMarket X OfferType intersection
```

```
        });
    }
});
```

Param `stockExchange` is object with shape:

```
{
  "baseCurrency": "PLN",
  "quoteCurrency": "LTC"
}
```

Example response:

```
[
    {
        aggregatedOffers: [
            {
                aggregationTime: 1535121440926,
                amount: "1.00000000",
                unitPrice: "30.00000000",
            },
                {…}, {…}, {…}, {…}, …
        ]
        baseCurrency: "BTC"
        quoteCurrency: "EUR"
        type: "buy"
    },
```

```
        {
        aggregatedOffers: [
            {
                aggregationTime: 1536293178182,
                amount: "0.01142228",
                unitPrice: "5700.00000000",
            },
                {…}, {…}, {…}, {…}, …
        ]
        baseCurrency: "BTC"
        quoteCurrency: "EUR"
        type: "sell"
    }

]
```

Transactions

It’s possible to build transactions list by:

- Listening to subsequent transactions
- Fetching N last transations

How to fetch N transactions from WebSocket API?

```
var socket = socketCluster.connect();

socket.emit('transactions', listOptions, function(errCode,
```

```
 response) {
    if (errCode) {
        // error occurred - details are in response
    } else {
        // success
        response.forEach(function(transactionDTO) {
            // handle single transaction
        });
    }
});
```

Important param `listOptions` have shape:

```
interface TransactionListOptions {
    "limit": number;
    "baseCurrency": string;
    "quoteCurrency": string;
}
```

All above fields are obligatory.