Websocket API

A high-level overview of working with 1inch Fusion+ orders.

Example

```
import { WebSocketApi, NetworkEnum } from "@1inch/fusion-sdk";

const wsSdk = new WebSocketApi({
  url: "wss://api.1inch.dev/fusion/ws",
  network: NetworkEnum.ETHEREUM,
  authKey: "your-auth-key",
});

wsSdk.order.onOrder((data) => {
  console.log("received order event", data);
});
```

Creation

Creation with a constructor

```
import { WebSocketApi, NetworkEnum } from "@1inch/fusion-sdk";

const ws = new WebSocketApi({
  url: "wss://api.1inch.dev/fusion/ws",
  network: NetworkEnum.ETHEREUM,
  authKey: "your-auth-key",
});
```

Creation with a custom provider

You can provide a custom provider for WebSocket. By default, 1inch uses ws library.

```
import { WsProviderConnector, WebSocketApi } from "@1inch/fusion-sdk";

class MyFancyProvider implements WsProviderConnector {
  // ... user implementation
}

const url = "wss://api.1inch.dev/fusion/ws/v2.0/1";
const provider = new MyFancyProvider({ url });

const wsSdk = new WebSocketApi(provider);
```

Creation with a new static method

```
import { WebSocketApi, NetworkEnum } from "@1inch/cross-chain-sdk";

const ws = WebSocketApi.new({
  url: "wss://api.1inch.dev/fusion/ws",
  network: NetworkEnum.ETHEREUM,
});
```

Creation with lazy initialization

By default, when you create an instance of `WebSocketApi`, it automatically opens a WebSocket connection, which might be a problem in some cases. To avoid this, you can enable lazy initializations.

```
import { WebSocketApi, NetworkEnum } from "@1inch/cross-chain-sdk";

const ws = new WebSocketApi({
  url: "wss://api.1inch.dev/fusion/ws",
  network: NetworkEnum.ETHEREUM,
  lazyInit: true,
});

ws.init();
```

Base methods

on

This method is used to subscribe to any event.

Arguments:

- [0] `event: string`
- [1] `cb: Function`

Example:

```
import { WebSocketApi, NetworkEnum } from "@1inch/fusion-sdk";

const ws = new WebSocketApi({
  url: "wss://api.1inch.dev/fusion/ws",
  network: NetworkEnum.ETHEREUM,
});

ws.on(WebSocketEvent.Error, console.error);

ws.on(WebSocketEvent.Open, function open() {
  ws.send("something");
});

ws.on(WebSocketEvent.Message, function message(data) {
  console.log("received: %s", data);
});
```

off

*This method is used to unsubscribe from any event.

Arguments:

- [0] `event: string`
- [1] `cb: Function`

Example:

```
import { WebSocketApi, NetworkEnum } from "@1inch/fusion-sdk";

const ws = new WebSocketApi({
  url: "wss://api.1inch.dev/fusion/ws",
  network: NetworkEnum.ETHEREUM,
});

ws.on(WebSocketEvent.Error, console.error);

ws.on(WebSocketEvent.Open, function open() {
  ws.send("something");
});

function message(data) {
  console.log("received: %s", data);
}

ws.on(WebSocketEvent.Message, message);

ws.off(WebSocketEvent.Message, message);
```

onOpen

This method is used to subscribe to an open event.

Arguments:

- [0] `cb: Function`

Example:

```
import { WebSocketApi, NetworkEnum } from "@1inch/fusion-sdk";

const ws = new WebSocketApi({
  url: "wss://api.1inch.dev/fusion/ws",
  network: NetworkEnum.ETHEREUM,
});

ws.onOpen(() => {
  console.log("connection is opened");
});
```

send

This method is used to send an event to backend.

Arguments:

- [0] `message`: any message which can be serialized with `JSON.stringify`

Example:

```
import { WebSocketApi, NetworkEnum } from "@1inch/fusion-sdk";

const ws = new WebSocketApi({
  url: "wss://api.1inch.dev/fusion/ws",
  network: NetworkEnum.ETHEREUM,
});

ws.send("my message");
```

close

Description: close connection

Example:

```
import { WebSocketApi, NetworkEnum } from "@1inch/fusion-sdk";

const ws = new WebSocketApi({
  url: "wss://api.1inch.dev/fusion/ws",
  network: NetworkEnum.ETHEREUM,
});

ws.close();
```

onMessage

This method is used to subscribe to a message event.

Arguments:

- [0] `cb: (data: any) => void`

Example:

```
import { WebSocketApi, NetworkEnum } from "@1inch/fusion-sdk";

const ws = new WebSocketApi({
  url: "wss://api.1inch.dev/fusion/ws",
  network: NetworkEnum.ETHEREUM,
});

ws.onMessage((data) => {
  console.log("message received", data);
});
```

onClose

This method is used to subscribe to a close event.

Example:

```
import { WebSocketApi, NetworkEnum } from "@1inch/fusion-sdk";

const ws = new WebSocketApi({
  url: "wss://api.1inch.dev/fusion/ws",
  network: NetworkEnum.ETHEREUM,
});

ws.onClose(() => {
  console.log("connection is closed");
});
```

onError

This method is used to subscribe to an error event.

Arguments:

- [0] `cb: (error: any) => void`

Example:

```
import { WebSocketApi, NetworkEnum } from "@1inch/fusion-sdk";

const ws = new WebSocketApi({
  url: "wss://api.1inch.dev/fusion/ws",
  network: NetworkEnum.ETHEREUM,
});
```

```
ws.onError((error) => {
  console.log("error is received", error);
});
```

Order namespace methods

onOrder

This method is used to subscribe to order events.

Arguments:

- [0] `cb: (data: OrderEventType) => void`

Example:

```
import { WebSocketApi, NetworkEnum } from "@1inch/fusion-sdk";

const ws = new WebSocketApi({
  url: "wss://api.1inch.dev/fusion/ws",
  network: NetworkEnum.ETHEREUM,
});

ws.order.onOrder((data) => {
  if (data.event === "order_created") {
    // do something
  }
  if (data.event === "order_invalid") {
    // do something
  }
});
```

onOrderCreated

This method is used to subscribe to the `order_created` events.

Arguments:

- [0] `cb: (data: OrderCreatedEvent) => void`

Example:

```
import { WebSocketApi, NetworkEnum } from "@1inch/fusion-sdk";

const ws = new WebSocketApi({
  url: "wss://api.1inch.dev/fusion/ws",
  network: NetworkEnum.ETHEREUM,
});

ws.order.onOrderCreated((data) => {
  // do something
});
```

onOrderInvalid

This method is used to subscribe to the `order_invalid` events.

Arguments:

- [0] `cb: (data: OrderInvalidEvent) => void`

Example:

```
import { WebSocketApi, NetworkEnum } from "@1inch/fusion-sdk";

const ws = new WebSocketApi({
  url: "wss://api.1inch.dev/fusion/ws",
  network: NetworkEnum.ETHEREUM,
});

ws.order.onOrderInvalid((data) => {
  // do something
});
```

onOrderBalanceOrAllowanceChange

This method is used to subscribe to the `order_balance_or_allowance_change` events.

Arguments:

- [0] `cb: (data: OrderBalanceOrAllowanceChangeEvent) => void`

Example:

```
import { WebSocketApi, NetworkEnum } from "@1inch/fusion-sdk";

const ws = new WebSocketApi({
  url: "wss://api.1inch.dev/fusion/ws",
  network: NetworkEnum.ETHEREUM,
});

ws.order.onOrderBalanceOrAllowanceChange((data) => {
  // do something
});
```

onOrderFilled

This method is used to subscribe to the `order_filled` events.

Arguments:

- [0] `cb: (data: OrderFilledEvent) => void`

Example:

```
import { WebSocketApi, NetworkEnum } from "@1inch/fusion-sdk";

const ws = new WebSocketApi({
  url: "wss://api.1inch.dev/fusion/ws",
  network: NetworkEnum.ETHEREUM,
});

ws.order.onOrderFilled((data) => {
  // do something
});
```

onOrderFilledPartially

This method is used to subscribe to the `order_filled_partially` events.

Arguments:

- [0] `cb: (data: OrderFilledPartiallyEvent) => void`

Example:

```
import { WebSocketApi, NetworkEnum } from "@1inch/fusion-sdk";

const ws = new WebSocketApi({
  url: "wss://api.1inch.dev/fusion/ws",
  network: NetworkEnum.ETHEREUM,
});

ws.order.onOrderFilledPartially((data) => {
  // do something
});
```

onOrderCancelled

This method is used to subscribe to the `order_cancelled` events.

Arguments:

- [0] `cb: (data: OrderCancelledEvent) => void`

Example:

```
import { WebSocketApi, NetworkEnum } from "@1inch/fusion-sdk";

const ws = new WebSocketApi({
  url: "wss://api.1inch.dev/fusion/ws",
  network: NetworkEnum.ETHEREUM,
});

ws.order.onOrderCancelled((data) => {
  // do something
});
```

RPC namespace methods

onPong

This method is used to subscribe to ping response.

Arguments:

- [0] `cb: (data: string) => void`

Example:

```
import { WebSocketApi, NetworkEnum } from "@1inch/fusion-sdk";

const ws = new WebSocketApi({
  url: "wss://api.1inch.dev/fusion/ws",
  network: NetworkEnum.ETHEREUM,
});

ws.rpc.onPong((data) => {
  // do something
});
```

ping

This method is used to ping healthcheck.

Example:

```
import { WebSocketApi, NetworkEnum } from "@1inch/fusion-sdk";

const ws = new WebSocketApi({
  url: "wss://api.1inch.dev/fusion/ws",
  network: NetworkEnum.ETHEREUM,
});

ws.rpc.ping();
```

getAllowedMethods

This method is used to get the list of allowed methods.

Example:

```
import { WebSocketApi, NetworkEnum } from "@1inch/fusion-sdk";

const ws = new WebSocketApi({
  url: "wss://api.1inch.dev/fusion/ws",
  network: NetworkEnum.ETHEREUM,
});

ws.rpc.getAllowedMethods();
```

onGetAllowedMethods

This method is used to subscribe to get the allowed methods response.

Arguments:

- [0] `cb: (data: RpcMethod[]) => void`

Example:

```
import { WebSocketApi, NetworkEnum } from "@1inch/fusion-sdk";

const ws = new WebSocketApi({
  url: "wss://api.1inch.dev/fusion/ws",
  network: NetworkEnum.ETHEREUM,
});

ws.rpc.onGetAllowedMethods((data) => {
  // do something
});
```

getActiveOrders

This method is used to get the list of active orders.

Example:

```
import { WebSocketApi, NetworkEnum } from "@1inch/fusion-sdk";

const ws = new WebSocketApi({
  url: "wss://api.1inch.dev/fusion/ws",
  network: NetworkEnum.ETHEREUM,
});

ws.rpc.getActiveOrders();
```

onGetActiveOrders

This method is used to subscribe to get active orders events.

Arguments:

- [0] `cb: (data: PaginationOutput\<ActiveOrder\>) => void`

Example:

```
import { WebSocketApi, NetworkEnum } from "@1inch/fusion-sdk";

const ws = new WebSocketApi({
  url: "wss://api.1inch.dev/fusion/ws",
  network: NetworkEnum.ETHEREUM,
});

ws.rpc.onGetActiveOrders((data) => {
  // do something
});
```

Types

OrderEventType

```
import { OrderType } from "./types";

type Event<K extends string, T> = { event: K; data: T };

export type OrderEventType =
  | OrderCreatedEvent
  | OrderInvalidEvent
  | OrderBalanceChangeEvent
  | OrderAllowanceChangeEvent
  | OrderFilledEvent
  | OrderFilledPartiallyEvent
  | OrderCancelledEvent
  | OrderSecretSharedEvent;

export enum EventType {
  OrderCreated = "order_created",
  OrderInvalid = "order_invalid",
  OrderBalanceChange = "order_balance_change",
  OrderAllowanceChange = "order_allowance_change",
  OrderFilled = "order_filled",
  OrderFilledPartially = "order_filled_partially",
  OrderCancelled = "order_cancelled",
  OrderSecretShared = "secret_shared",
}

type OrderCreatedEvent = Event<
  "order_created",
  {
    orderHash: string;
    signature: string;
    order: LimitOrderV3Struct;
    deadline: string;
    auctionStartDate: string;
    auctionEndDate: string;
    remainingMakerAmount: string;
  }
>;

export type OrderCreatedEvent = Event<
  EventType.OrderCreated,
  {
    srcChainId: SupportedChain;
    dstChainId: SupportedChain;
    orderHash: string;
    order: LimitOrderV4Struct;
    extension: string;
    signature: string;
    isMakerContract: boolean;
    quoteId: string;
    merkleLeaves: string[];
    secretHashes: string[];
  }
>;

export type OrderBalanceChangeEvent = Event<
  EventType.OrderBalanceChange,
  {
    orderHash: string;
    remainingMakerAmount: string;
    balance: string;
  }
>;

export type OrderAllowanceChangeEvent = Event<
  EventType.OrderAllowanceChange,
  {
    orderHash: string;
    remainingMakerAmount: string;
    allowance: string;
  }
>;

type OrderInvalidEvent = Event<
  EventType.OrderInvalid,
  {
    orderHash: string;
  }
>;

export type OrderCancelledEvent = Event<
  EventType.OrderCancelled,
  {
    orderHash: string;
    remainingMakerAmount: string;
  }
>;

type OrderFilledEvent = Event<EventType.OrderFilled, { orderHash: string }>;

type OrderFilledPartiallyEvent = Event<
  EventType.OrderFilledPartially,
    { orderHash: string; remainingMakerAmount: string }
>;

export type OrderSecretSharedEvent = Event<
  EventType.OrderSecretShared,
  {
    idx: number;
    secret: string;
    srcImmutables: Jsonify<Immutables>;
    dstImmutables: Jsonify<Immutables>;
  }
>;
```

RpcMethod

```
export enum RpcMethod {
  GetAllowedMethods = "getAllowedMethods",
  Ping = "ping",
  GetActiveOrders = "getActiveOrders",
  GetSecrets = "getSecrets",
}
```

Previous
‹ Auction suffix

Next
Introduction ›