SDK Overview

This example provides high level functionality to working with the 1inch Fusion+ SDK:

Initialization

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

async function main() {
  const sdk = new SDK({
    url: "https://api.1inch.dev/fusion-plus",
    authKey: "your-auth-key",
  });

  const orders = await sdk.getActiveOrders({ page: 1, limit: 2 });
}

main();
```

Creation

Constructor arguments params: CrossChainSDKConfigParams

```
interface HttpProviderConnector {
  get<T>(url: string): Promise<T>;

  post<T>(url: string, data: unknown): Promise<T>;
}

interface BlockchainProviderConnector {
  signTypedData(
    walletAddress: string,
    typedData: EIP712TypedData,
  ): Promise<string>;

  ethCall(contractAddress: string, callData: string): Promise<string>;
}

type CrossChainSDKConfigParams = {
  url: string;
  blockchainProvider?: BlockchainProviderConnector;
  httpProvider?: HttpProviderConnector; // by default we are using axios
};
```

Example with custom httpProvider

```
import { api } from "my-api-lib";

class CustomHttpProvider implements HttpProviderConnector {
  get<T>(url: string): Promise<T> {
    return api.get(url);
  }

  post<T>(url: string, data: unknown): Promise<T> {
    return api.post(url, data);
  }
}
```

Methods

getActiveOrders

Description: used to get the list of active orders **Arguments:**

- [0] PaginationParams

Example

```
import { SDK, NetworkEnum } from "@1inch/cross-chain-sdk";
const sdk = new SDK({
  url: "https://api.1inch.dev/fusion-plus",
  authKey: "your-auth-key",
});
const orders = await sdk.getActiveOrders({ page: 1, limit: 2 });
```

getOrdersByMaker

Description used to get orders by maker

Arguments

- [0] params: PaginationParams & {address: string}

Example

```
import { SDK, NetworkEnum } from "@1inch/cross-chain-sdk";
const sdk = new FusionSDK({
  url: "https://api.1inch.dev/fusion-plus",
  authKey: "your-auth-key",
});
```

```
const orders = await sdk.getOrdersByMaker({
  page: 1,
  limit: 2,
  address: "0xfa80cd9b3becc0b4403b0f421384724f2810775f",
});
```

getQuote

Description: Get quote details based on input data

Arguments:

- [0] params: QuoteParams

Example:

```
import { SDK, NetworkEnum, QuoteParams } from "@1inch/cross-chain-sdk";
const sdk = new FusionSDK({
  url: "https://api.1inch.dev/fusion-plus",
  authKey: "your-auth-key",
});

const params = {
  srcChainId: NetworkEnum.ETHEREUM,
  dstChainId: NetworkEnum.GNOSIS,
  srcTokenAddress: "0x6b175474e89094c44da98b954eedeac495271d0f",
  dstTokenAddress: "0xeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee",
  amount: "1000000000000000000000",
  walletAddress: "0x123....",
};

const quote = await sdk.getQuote(params);
```

createOrder

Description: used to create a cross chain order by given quote

Arguments:

- [0] quote: Quote
- [1] params: OrderParams

Example:</b<

```
import {
  getRandomBytes32,
  SDK,
  HashLock,
  PrivateKeyProviderConnector,
  NetworkEnum,
} from "@1inch/cross-chain-sdk";

const makerPrivateKey = "0x123....";
const makerAddress = "0x123....";

const nodeUrl = "....";

const blockchainProvider = new PrivateKeyProviderConnector(
  makerPrivateKey,
  new Web3(nodeUrl),
);

const sdk = new SDK({
  url: "https://api.1inch.dev/fusion-plus",
  authKey: "your-auth-key",
  blockchainProvider,
});

const params = {
  srcChainId: NetworkEnum.ETHEREUM,
  dstChainId: NetworkEnum.GNOSIS,
  srcTokenAddress: "0x6b175474e89094c44da98b954eedeac495271d0f",
  dstTokenAddress: "0xeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee",
  amount: "1000000000000000000000",
  enableEstimate: true,
  walletAddress: makerAddress,
};

const quote = await sdk.getQuote(params);

const secretsCount = quote.getPreset().secretsCount;

const secrets = Array.from({ length: secretsCount }).map(() =>
  getRandomBytes32(),
);
const secretHashes = secrets.map((x) => HashLock.hashSecret(x));

const hashLock =
  secretsCount === 1
    ? HashLock.forSingleFill(secrets[0])
    : HashLock.forMultipleFills(
        secretHashes.map((secretHash, i) =>
          solidityPackedKeccak256(
            ["uint64", "bytes32"],
            [i, secretHash.toString()],
          ),
        ) as (string & {
          _tag: "MerkleLeaf";
        })[],
      );

sdk
  .createOrder(quote, {
    walletAddress: makerAddress,
    hashLock,
    secretHashes,
    // fee is an optional field
    fee: {
      takingFeeBps: 100, // 1% as we use bps format, 1% is equal to 100bps
      takingFeeReceiver: "0x0000000000000000000000000000000000000000", //  fee receiver address
    },
  })
  .then(console.log);
```

Types

PaginationParams

```
type PaginationParams = {
  page?: number; // default is 1
  limit?: number; // default is 2, min is 1, max is 500
};
```

QuoteParams

```
type QuoteParams = {
  fromTokenAddress: string;
  toTokenAddress: string;
  amount: string;
  permit?: string; // a permit (EIP-2612) call data, user approval sign
  takingFeeBps?: number; // 100 == 1%
};
```

OrderParams

```
enum PresetEnum {
  fast = "fast",
  medium = "medium",
  slow = "slow",
}

type OrderParams = {
  fromTokenAddress: string;
  toTokenAddress: string;
  amount: string;
  walletAddress: string;
  permit?: string; // a permit (EIP-2612) call data, user approval sign
  receiver?: string; // address
  preset?: PresetEnum;
  nonce?: OrderNonce | string | number; // allows to batch cancel orders. by default: not used
  fee?: TakingFeeInfo;
};

export type TakingFeeInfo = {
  takingFeeBps: number; // 100 == 1%
  takingFeeReceiver: string;
};
```

Previous

Next
When and how to submit a secret ›