Auction calculator

This method is used to calculate taker amount and auction rate.

Example

```
import { AuctionCalculator } from "@1inch/fusion-sdk";

const limitOrderStruct = {
  allowedSender: "0x0000000000000000000000000000000000000000",
  interactions:
    "0x000c004e200000000000000000219ab540356cbb839cbe05303d7705faf486570009",
  maker: "0x00000000219ab540356cbb839cbe05303d7705fa",
  makerAsset: "0xc02aaa39b223fe8d0a0e5c4f27ead9083c756cc2",
  makingAmount: "1000000000000000000",
  offsets: "0",
  receiver: "0x0000000000000000000000000000000000000000",
  salt: "45118768841948961586167738353692277076075522015101619148498725069326976558864",
  takerAsset: "0xa0b86991c6218b36c1d19d4a2e9eb0ce3606eb48",
  takingAmount: "1420000000",
};

const calculator = AuctionCalculator.fromLimitOrderV3Struct(limitOrderStruct);
// #=> AuctionCalculator instance

const rate = calculator.calcRateBump(1673548209);
// #=> 14285

const auctionTakingAmount = calculator.calcAuctionTakingAmount(
  "1420000000",
  rate,
);
// #=> '1422028470'
```

static AuctionCalculator.fromLimitOrderV3Struct

This method is used to create an auction instance from a limit order.

Arguments:

- `LimitOrderV3Struct`

Name	Type	Inner Solidity type	Description
salt	string	uint256	Some unique value. It is necessary to be able to create limit orders with the same parameters (so that they have a different hash).
makerAsset	string	address	The address of the asset user wants to sell (address of a token contract).
takerAsset	string	address	The address of the asset user wants to buy (address of a token contract).
maker	string	address	The address of the limit order creator.
receiver	string	address	If it contains a zero address, it means that taker asset will be sent to the address of the creator of the limit order. If a user set any other value, then taker asset will be sent to the specified address.
allowedSender	string	address	If it contains a zero address, it means that a limit order is available for everyone to fill. If a user set any other value, then the limit order will be available for execution only for the specified address (private limit order).
makingAmount	string	uint256	Amount of maker asset.
takingAmount	string	uint256	Amount of taker asset.
offsets	string	uint256	Every 32's bytes represents offset of the n'ths interaction.
interactions	string	bytes	Used to encode Fusion-specific data.

Order interactions suffix structure:

- M*(1 + 3 bytes) - auction points coefficients with seconds delays.
- N*(4 + 20 bytes) - resolver with corresponding time limit.
- 4 bytes - public time limit (starting from this point of time, an order can be fullfilled by anyone).
- 32 bytes - taking fee (optional `if` flags have `\_HAS_TAKING_FEE_FLAG`).
- 1 byte - flags.

Example:

```
import { AuctionCalculator } from "@1inch/fusion-sdk";

const limitOrderStruct = {
  allowedSender: "0x0000000000000000000000000000000000000000",
  interactions:
    "0x000c004e200000000000000000219ab540356cbb839cbe05303d7705faf486570009",
  maker: "0x00000000219ab540356cbb839cbe05303d7705fa",
  makerAsset: "0xc02aaa39b223fe8d0a0e5c4f27ead9083c756cc2",
  makingAmount: "1000000000000000000",
  offsets: "0",
  receiver: "0x0000000000000000000000000000000000000000",
  salt: "45118768841948961586167738353692277076075522015101619148498725069326976558864",
  takerAsset: "0xa0b86991c6218b36c1d19d4a2e9eb0ce3606eb48",
  takingAmount: "1420000000",
};

AuctionCalculator.fromLimitOrderV3Struct(limitOrderStruct);
// #=> AuctionCalculator instance
```

AuctionCalculator.calcRateBump

This method is used to calculate exchange rate at some point of time. To learn more about it, read Fusion swap: Introduction.

Arguments:

- `time` (Unix timestamp)

AuctionCalculator.calcAuctionTakingAmount

This method is used to calculate taker amount.

Arguments:

- `takingAmount: string`
- `rate: number`

AuctionCalculator.fromAuctionData

This method is used to create `AuctionCalculator` from suffix and salt.

Arguments:

- `suffix`: AuctionSuffix
- `salt`: AuctionSalt

Example:

```
import {
  AuctionSuffix,
  AuctionSalt,
  AuctionCalculator,
} from "@1inch/fusion-sdk";

const suffix = AuctionSuffix.decode(
  "0x000c004e200000000000000000219ab540356cbb839cbe05303d7705faf486570009",
);
const salt = AuctionSalt.decode(
  "45118768841948961586167738353692277076075522015101619148498725069326976558864",
);
AuctionCalculator.fromAuctionData(suffix, salt);
// #=> AuctionCalculator instance
```

Previous

When and how to submit a secret

Next

Auction salt