

Interaction receiver

Limit order protocol has the ability to call an intermediate contract when an order is filled.

Intermediate contract should implement an interface: [InteractiveNotificationReceiver.sol](#)

Use case

For example, we want to create a limit order 1 INCH - WETH, but so that when this order is filled, we will not receive WETH but ETH.

To do this, let's create a [smart contract](#) :

```
contract
```

```
WethUnwrapper
```

```
is InteractiveNotificationReceiver { // solhint-disable-next-line no-empty-blocks receive ( )
```

```
external
```

```
payable
```

```
{ }
```

```
function
```

```
notifyFillOrder ( address
```

```
/ taker / , address
```

```
/ makerAsset / , address takerAsset , uint256
```

```
/ makingAmount / , uint256 takingAmount , bytes
```

```
calldata interactiveData )
```

```
external override { address
```

```
payable makerAddress ; // solhint-disable-next-line no-inline-assembly assembly
```

```
{ makerAddress :=
```

```
shr ( 96 ,
```

```
calldataload ( interactiveData . offset ) ) } IWithdrawable ( takerAsset ) . withdraw ( takingAmount ) ; makerAddress . transfer ( takingAmount ) ; } } And create a limit order with preInteraction and/or postInteraction field:
```

```
const interactiveReceiverAddress =
```

```
'0x1282d0c06368c40c8d4a4d818d78f258d982437b' ; const walletAddress =
```

```
'0xfb3c7ebccccAA12B5A884d612393969Aaaaaaaaa' ;
```

```
const limitOrder = limitOrderBuilder . buildLimitOrder ( { makerAssetAddress :
```

```
'0xbb4c9cb9cb36b01bd1c8aebf2de08d9173bc095c' , takerAssetAddress :
```

```
'0x111111111117dc0aa78b770fa6a738034120c302' , makerAddress : walletAddress , receiver : interactiveReceiverAddress , makerAmount :
```

```
'100' , takerAmount :
```

```
'200' ,
```

```
// What to do before the transfer preInteraction :
```

```
'0x' , // What to do after the transfer has been made postInteraction : interactiveReceiverAddress + walletAddress . slice ( 2 ) , } } ; As you can see, the postInteraction field consists of two parts - the contract address and the interactiveData (which contains the wallet address). We also set the interactiveReceiverAddress as the receiver . This is necessary in order for this contract to receive a WETH when filling out a limit order and return ETH to the creator of the limit order. Now the limit order
```

will work according to the following scenario:

- someone (taker
-) will create a transaction to execute a limit order
- taker
- funds (WETH
-) will be transferred to the interactiveReceiver
- contract
- interactiveReceiver
- contract will send the same amount of ETH
- to the maker
- and finally, the maker's asset
- (1 INCH
-) will be transferred from the maker's
- address to the taker's
- address

[Example of transaction](#)

However, this is one example of the use of interactivity. You can implement any other scenario [Edit this page](#) [Previous](#) [Validate a limit order](#) [Next](#) [Domain separator](#)