

# Transaction

(go-ethereum version 1.8)

[core/types/transaction.go]

```
type txdata struct {  
    AccountNonce uint64           `json:"nonce" gencodec:"required"`  
    Price         *big.Int         `json:"gasPrice" gencodec:"required"`  
    GasLimit      uint64           `json:"gas" gencodec:"required"`  
    Recipient     *common.Address `json:"to" rlp:"nil" gencodec:"required" // nil means contract`  
  
    creation  
    Amount        *big.Int         `json:"value" gencodec:"required"`  
    Payload       []byte          `json:"input" gencodec:"required"`  
  
    // Signature values  
    V *big.Int `json:"v" gencodec:"required"`  
    R *big.Int `json:"r" gencodec:"required"`  
    S *big.Int `json:"s" gencodec:"required"`  
  
    // This is only used when marshaling to JSON.  
    Hash *common.Hash `json:"hash" rlp:"-"  
}
```

Transaction

to address

value (Wei)

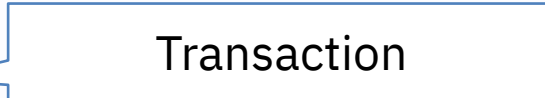
input data

# Transaction Ordering Examples

[github.com/sambacha/mev-slides](https://github.com/sambacha/mev-slides)


## Naive approach implementation examples (FIXME)

```
//approach 1
transactions := self.eth.TxPool().GetTransactions()
sort.Sort(types.TxByNonce{transactions})
```




Transaction

```
//approach 2
transactions := self.eth.TxPool().GetTransactions()
sort.Sort(types.TxByPriceAndNonce{transactions})
```




order by price & nonce

```
// approach 3
// commit transactions for this run.
txPerOwner := make(map[common.Address]types.Transactions)
// Sort transactions by owner
for _, tx := range self.eth.TxPool().GetTransactions() {
```



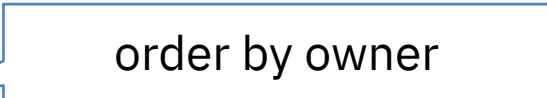
sort by owner

```
    from, _ := tx.From() // we can ignore the sender error
    txPerOwner[from] = append(txPerOwner[from], tx)
}
var (
    singleTxOwner types.Transactions
    multiTxOwner  types.Transactions
)
```



input data

```
// Categorize transactions by
// 1. 1 owner tx per block
// 2. multi txs owner per block
```



order by owner

```
for _, txs := range txPerOwner {
    if len(txs) == 1 {
        singleTxOwner = append(singleTxOwner, txs[0])
    } else {
        multiTxOwner = append(multiTxOwner, txs...)
    }
}
sort.Sort(types.TxByPrice{singleTxOwner})
sort.Sort(types.TxByNonce{multiTxOwner})
transactions := append(singleTxOwner, multiTxOwner...)
```