

The Smart Contract maintains a couple of variables. Below is the information for the same.

//holds the address of all the registered auctioneers

**address[] auctioneer;**

//boolean variable to hold of registration process is on

**bool isRegistrationStopped;**

// boolean variable to know if bidding is on or of

**bool bidStopped;**

// holds the map of address and the committed value

**mapping(address => bytes32)**

**public commitment;**

//keys for commitment map

**address[] private  
commitmentKeys;**

//map of address and bid in use

**mapping(address => uint)  
public bidMap;  
address[] private bidMapKeys;**

// list of all valid auctioneers

**address payable[]  
validAuctioneer;**

//owner of the smart contract

**address owner;**

//map to keep note of the money  
locked which needs to be returned

```
    mapping(address => uint)
private auctioneerLockedEther;
    address[] private
auctioneerLockedEtherKeys;
```

```
    //list of auction_winners
    mapping(uint => address)
auction_winners;
    uint auction_number;
    // exchange rate
    uint public ETHUSD;
```

## **Additional useful functions**

**getWinners:** get list of winners

**getBalance:** get balance of smart contract

Test Case Information below

5 users (User 5 and user 4 are  
invalid users)

User 4 places less value in  
commitment and user 5 hash  
doesn't match case

1st User

Address:

0xdA5EC4f3e814EDD335Fe7154  
C11BC2Bd4e622B97

Bid: 190 USD

Random\_seed: 1

Hash for 1st:

0xe9e5335a6df74f7c756d3a5dc5b9613678  
cac204064bc8117bc641392424615a

**Commitment: 5 ethers**

2nd User

Address:

0xF8b81688544617FEdfC9476e1  
24a94Ef3669Da62

Bid: 380 USD

Random\_seed: 2

Hash for 2nd:

0x21dc236545efd2fc4351378dff99f054fc3  
71432ee335aa3d3f93d1d9b0d0d34

**Commitment: 5 ethers**

3rd User - 5

Address:

0x2F964B3815269eA062979b22C  
75534A4B2eb12Ab

Bid: 570 USD

Random\_seed: 3

Hash for 1st:

0x941ee0bd7e9c431dd1eab2acbce65c

9d4d7baad0af9fd527dfba10c5e82a1c5  
d

**Commitment: 5 ethers**

## **Less Commitment case**

4th User - 2

Address:

0x0Ede2B276B7db4f6D572908FF  
5daAb3270A30dc2

Bid: 760 USD

Random\_seed: 4

Hash:

0x24ad6262452c5d2fed5d01d15e9bf  
77414c7c97d131aab1a802d1b6a3c3d  
b22a

**Commitment: 2 ethers**

# Hash Mismatch Case

**5th User - 15**

**Address:**

**0xfbaA5E377f5E36c41b2fF15ED  
D725C45750Ef7e5**

**Bid: 1900**

**Random\_seed: 5**

**Hash:**

**0x5faa094ed6f0a1b4e26191a332d  
c5b71512e61353ebb19bae953aed  
5c828057f**

**Expected Results**

**Winner - User 3 (Address:**

0x2F964B3815269eA062979b22C  
75534A4B2eb12Ab)  
Smart Contract Balance After  
finish: 20 ethers

Logs below:

transact to Auction.computeWinner pending ...

```
[vm]  
from:0x054...cbdd1  
to:Auction.computeWinner() 0x167...0924d  
value:0 wei  
data:0x9d9...d3b2d  
logs:0  
hash:0x528...e7446
```

Debug

<b>status</b>	0x1 Transaction mined and execution succeed
<b>transaction hash</b>	0x528ede2ff73b03e4e8a0509e00d6a5af34cb01c75b7e9eff6 46178abc5be7446
<b>from</b>	0x05409d15db2d368b922bb00547ad1706df4cbdd1
<b>to</b>	Auction.computeWinner() 0x1679591c985bde2415c7c8faa2cd82798680924d
<b>gas</b>	30000000000 gas
<b>transaction cost</b>	115319 gas
<b>execution cost</b>	94047 gas
<b>hash</b>	0x528ede2ff73b03e4e8a0509e00d6a5af34cb01c75b7e9eff6 46178abc5be7446



<b>input</b>	0x9d9...d3b2d
<b>decoded input</b>	{}
<b>decoded output</b>	{ "0": "address: 0x2F964B3815269eA062979b22C75534A4B2eb12Ab" }
<b>logs</b>	[]
<b>value</b>	0 wei

transact to Auction.getBalance pending ...

[vm]

**from:**0x054...cbdd1

**to:**Auction.getBalance() 0x167...0924d

**value:**0 wei

**data:**0x120...65fe0

**logs:**0

**hash:**0x8bb...9e47b

Debug

<b>status</b>	0x1 Transaction mined and execution succeed
<b>transaction hash</b>	0x8bbd8ab1706cf576f0f16f8883c59d59f5d6e19866998820f f40aee48399e47b
<b>from</b>	0x05409d15db2d368b922bb00547ad1706df4cbdd1

```
to          Auction.getBalance()  
           0x1679591c985bde2415c7c8faa2cd82798680924d  
  
gas         300000000000 gas  
  
transaction cost 21511 gas  
  
execution cost  239 gas  
  
hash         0x8bbd8ab1706cf576f0f16f8883c59d59f5d6e19866998820f  
           f40aee48399e47b  
  
input       0x120...65fe0  
  
decoded input {}  
  
decoded output { "0": "uint256: 20031914893617021276" }  
  
logs        []  
  
value       0 wei
```

>

**Below are the  
screenshots of  
various scenarios**

# #1 Deployment

The screenshot displays the Source Verifier web application interface. On the left sidebar, the 'ENVIRONMENT' section shows 'JavaScript VM' selected. The 'ACCOUNT' section displays '0x054...CBd1' with a balance of '99.999999999940377255 ether'. The 'GAS LIMIT' is set to '30000000000'. The 'VALUE' is '0' and the currency is 'ether'. The 'CONTRACT' section shows 'Auction - browser/Auction.sol' with a 'Deploy' button. Below this, there are options to 'PUBLISH TO IPFS' or 'At Address'. The 'Transactions recorded' section shows '314'. The 'Deployed Contracts' section lists 'HASHCALCULATOR AT 0x0B...D0500 (MEMORY)' and 'AUCTION AT 0x1A...D0368 (MEMORY)'. The 'AUCTION AT 0x1A...D0368 (MEMORY)' section shows a list of functions: '...callBack', '...callBack', 'commitBid', 'commitBidding', 'computePrice', 'getBalance', 'getBidUSD', 'registerBid', 'sanctified', 'registerAuctioneer', and 'resetAuction'. The main area shows the Solidity code for 'Auction.sol' with the following content:

```
1 pragma solidity ^0.5.11;
2
3 import "github.com/oraclize/ethereum-api/provableAPI.sol";
4 contract owned is usingProvable {
5     constructor() public { owner = msg.sender; }
6     address payable owner;
7
8     modifier onlyOwner {
9         require(
10             msg.sender == owner,
11             "Only owner can call this function."
12         );
13     };
14 }
15
```

Below the code, the 'creation of Auction pending...' section shows a transaction status. The transaction hash is '0x30933464360201f0e47f940c946c884da5170d15a80b16a19ab029207'. The transaction status is 'success'. The transaction cost is '385773 gas'. The execution cost is '251397 gas'. The hash is '0x30933464360201f0e47f940c946c884da5170d15a80b16a19ab029207'. The input is '0x054...82365'. The decoded input is '()'. The decoded output is '-'. The logs are '["From": "0x12f0e47f940c946c884da5170d15a80b16a19ab029207", "to": "0x1A...D0368", "value": "0 ether", "gas": "385773", "data": "0x054...82365"}']'. The logs are also shown in a table format.

## #2 registration of users

The image shows the Remix IDE interface. On the left, the 'RUN TRANSACTIONS' panel is active, displaying the account '0xdA5...22B97' with 98.989361702121117057 ether. The contract 'Auction - browser/Auction.sol' is selected. The 'Deploy' button is highlighted. Below the deploy button, there are buttons for 'At Address' and 'Load contract from Address'. The 'Transactions recorded' section shows 'Deployed Contracts' and a list of functions: 'callback', 'callback', 'commitBid', 'commitBidStop', 'computeWinner', 'getBalance', 'getBidUSD', 'registerStop', 'sendBid', 'registerAuctioneer', and 'resetAuction'. The 'Auction AT 0x1A1...08368 (MEMORY)' section is expanded, showing a list of functions: 'callback', 'callback', 'commitBid', 'commitBidStop', 'computeWinner', 'getBalance', 'getBidUSD', 'registerStop', 'sendBid', 'registerAuctioneer', and 'resetAuction'.

On the right, the 'HashCalculator.sol' file is open, showing the Solidity code for the 'Auction' contract. The code is as follows:

```
1 pragma solidity ^0.5.11;
2
3 import "github.com/oraize/ethereum-api/provableAPI.sol";
4 contract owned is usingProvable {
5     constructor() public { owner = msg.sender; }
6     address payable owner;
7
8     modifier onlyOwner {
9         require(
10             msg.sender == owner,
11             "Only owner can call this function."
12         );
13     }
14 }
15
```

Below the code, the 'Transaction to Auction.registerAuctioneer' is shown as pending. The transaction details are displayed in the bottom right panel:

State	Del Transaction mined and execution succeed
transaction hash	0x3b9e1a5e136d7d673443854b54436887f728b1c34763
from	0xdA5...22B97
to	Auction.registerAuctioneer() 0x1A1...08368
gas	3000000000 gas
transaction cost	45018 gas
execution cost	43744 gas
hash	0x3b9e1a5e136d7d673443854b54436887f728b1c34763
input	0x432...43512
decoded input	()
decoded output	()
logs	()
value	0 wei

## #3 Registration Error after its being stopped

**DEPLOY & RUN TRANSACTIONS**

At Address  OR

Transactions recorded **317**

Deployed Contracts

HASHCALCULATOR AT 0x2CB...0D500 (MEMORY)

▼ AUCTION AT 0x1A1...06368 (MEMORY)

- ...callBack  bytes32 mppl, string result
- ...callBack  bytes32 \_mppl, string \_result, bytes \_proof
- commitBid  bytes32 hash\_commit
- commitBidStop
- computeWinner
- getBalance
- getETHUSD
- registerStop
- sendBid  uint256 bid\_in\_1503, uint256 random\_seed, address public\_key
- registerAuctioneer
- resetAuction
- updatePrice
- bidMap  address
- commitment  address
- ETHUSD

Low level interactions

CALldata

```
1 pragma solidity ^0.5.11;
2
3 import "github.com/oraclize/ethereum-api/provableAPI.sol";
4 contract owned is usingProvable {
5     constructor() public { owner = msg.sender; }
6     address payable owner;
7
8     modifier onlyOwner {
9         require(
10             msg.sender == owner,
11             "Only owner can call this function."
12         );
13     }
14 }
15
```

transaction console

Search with transaction hash or address

transaction hash	gas
execution must	43748 gas
hash	0x38b0a10116d7d47364a0b5a0d77d34428258b5463b6875738a1a206765
input	0x432...42512
decoded input	()
decoded output	()
logs	()
value	0 wei

transact to Auction.registerStop pending ...

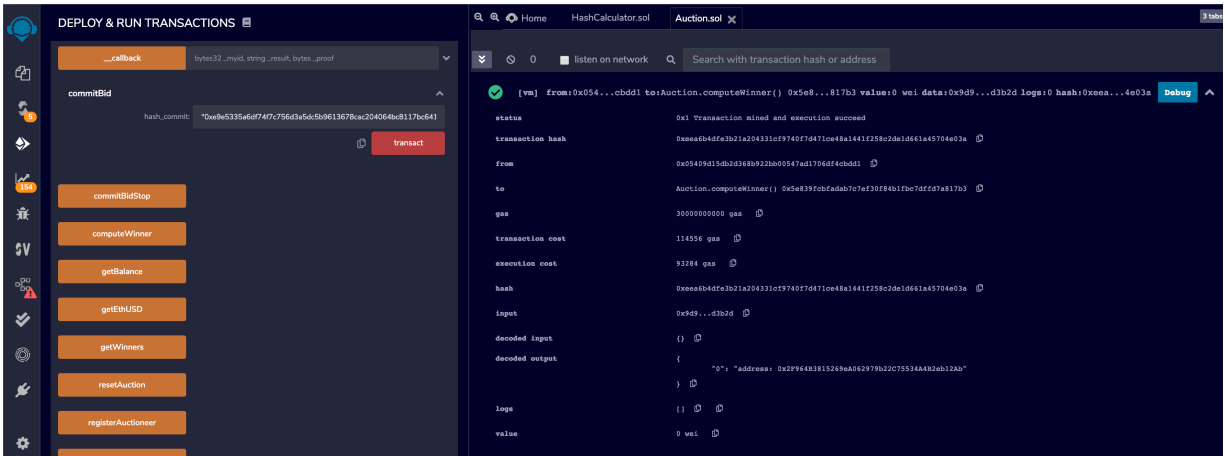
[vm] from:0x054...cbdd1 to:Auction.registerStop() 0x1a1...06368 value:0 wei data:0x2ba...02e8b logs:0 hash:0xae8...dd542

transact to Auction.registerStop pending ...

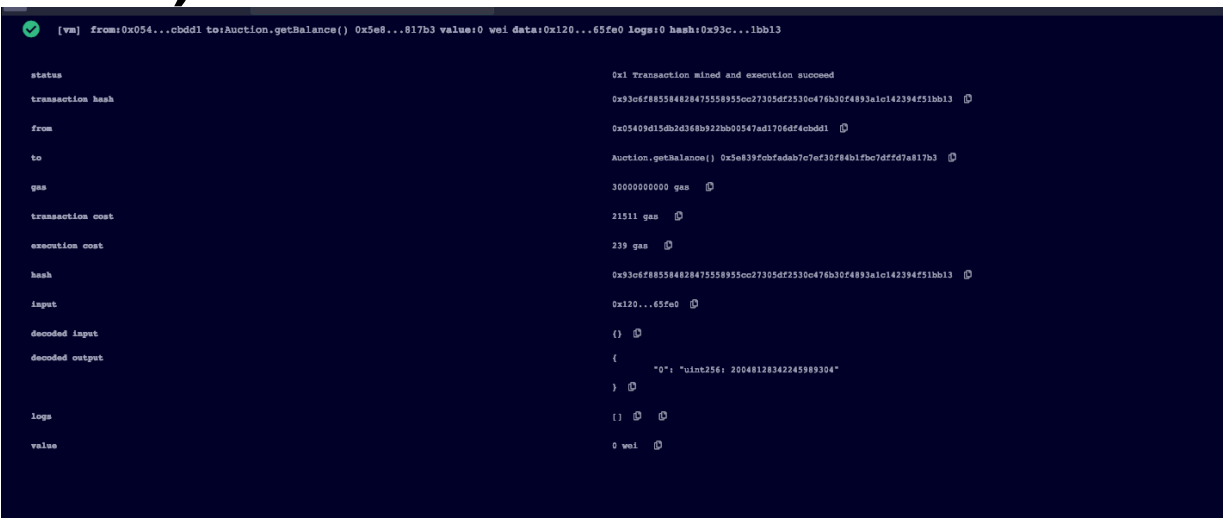
[vm] from:0xf8b...9da62 to:Auction.registerStop() 0x1a1...06368 value:0 wei data:0x2ba...02e8b logs:0 hash:0x9ef...d2a5e

transact to Auction.registerStop errored: VM error: revert.  
revert The transaction has been reverted to the initial state.  
Reason provided by the contract: "only owner can call this function.". Debug the transaction to get more information.

## #4 Compute Winner in above Test Case Scenario with 5 users 3rd User Wins the Auction



**Smart Contract Balance after  
Auction (value of auction winner (3  
ether) + value sent by 4th user (2  
ether) + value sent by 5th user (15  
ether) = 20 ether**



# getWinners() output for a use case

DEPLOY & RUN TRANSACTIONS

HASH-CALCULATOR AT 0x3C...0090 (MEMORY)

AUCTION AT 0x38B...BC93 (MEMORY)

\_\_callback

txns2\_myel\_dlog\_react

\_\_callback

txns2\_myel\_dlog\_react\_bake\_proof

commitBid

hash\_commit

0x411eaf8af7db411d41eab2a3eaf6c94479eae5af946274fba15d4

transact

commitBidLog

computeWinner

getBalance

getEthUSD

getWinners

resetAuction

registerAuctioneer

registerStop

sendBid

"130"/"11"0a48C4d4814D0339F7154C113C28a42289"

getWinners

bidMap

address

commitment

address

ETHUSD

winners

address

Low level interactions

CALL DATA

Transact

HashCalculator.sol Auction.sol

59 // send a query to update the price of Ether  
60  
61 function updatePrice() payable public {  
62 if (provable\_getPrice("URL") > address(this).balance) {  
63 emit NewProvableQuery("Provable query was NOT sent, please add some ETH to cover for  
64 } else {  
65 emit NewProvableQuery("Provable query was sent, standing by for the answer..");  
66 provable\_query("URL", "json(https://api.pro.coinbase.com/products/ETH-USD/ticker).pri  
67 }  
68  
69  
70  
71 function getBalance() public returns (uint) {  
72 return address(this).balance;  
73 }

listen on network Search with transaction hash or address

transact to Auction.getWinners pending ...

(wei) from:0x54...c0dd1 to:Auction.getWinners() 0x5b6...bc953 value:0 wei data:0x01...5c37e logs:0 hash:0x27...88006

status

tx: Transaction mined and execution returned

transaction hash

0x27354f5ed54e780298b4b3e9c8f86a7316c3e0f4d3756a54793088006

from

0x54549d15db3d38b720b07547d17040f4e0dd1

to

Auction.getWinners() 0x5b616a8ba75b35a304457431d13b5240be953

gas

3000000000 gas

transaction cost

24470 gas

execution cost

3198 gas

hash

0x27354f5ed54e780298b4b3e9c8f86a7316c3e0f4d3756a54793088006

logs

0x01...0d37e

decoded logs

(1)

decoded output

{  
 "P": "address"; 0x37946d381328ba361379b2070534a43ab12ab"  
}

logs

(1)

value

0 wei