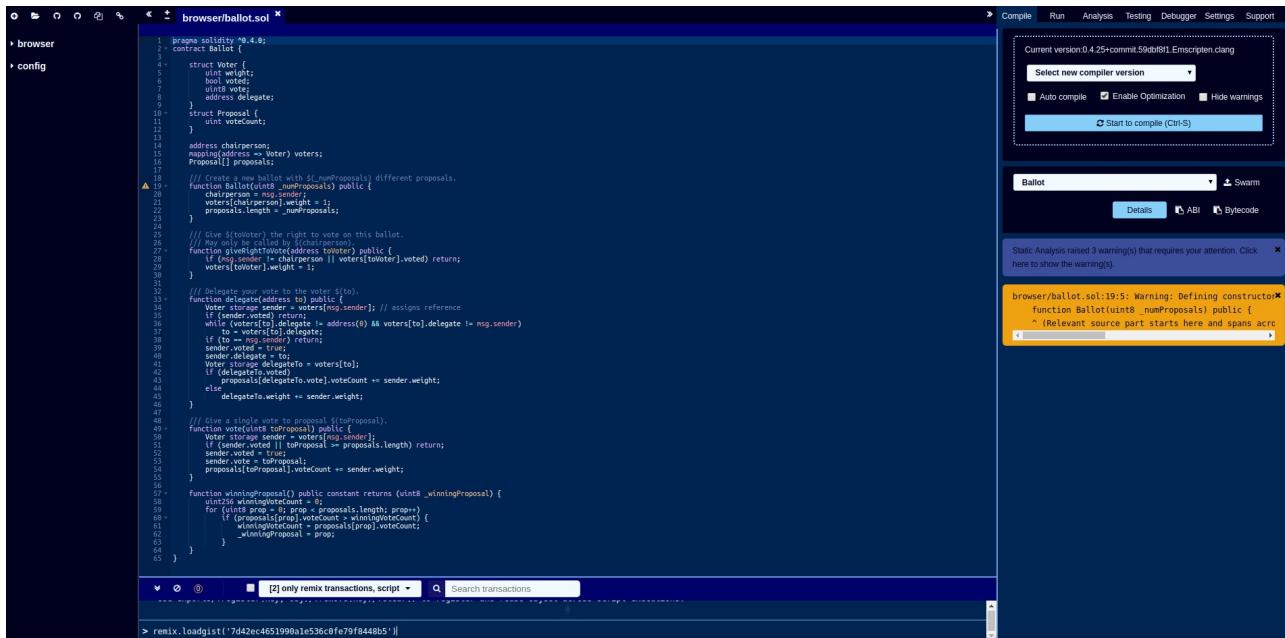


Presentation (record and replay contract deployment and interaction)

- gist 7d42ec4651990a1e536c0fe79f8448b5

0. load gist with files from terminal



```
pragma solidity >=0.4.0;
contract Ballot {
    struct Voter {
        uint weight;
        bool voted;
        uint proposal;
        address delegate;
    }
    struct Proposal {
        uint voteCount;
    }
    address chairperson;
    mapping(address => Voter) voters;
    Proposal[] proposals;
}

function Ballot(uint8 numProposals) public {
    chairperson = msg.sender;
    voters[msg.sender].weight = 1;
    proposals.length = numProposals;
}

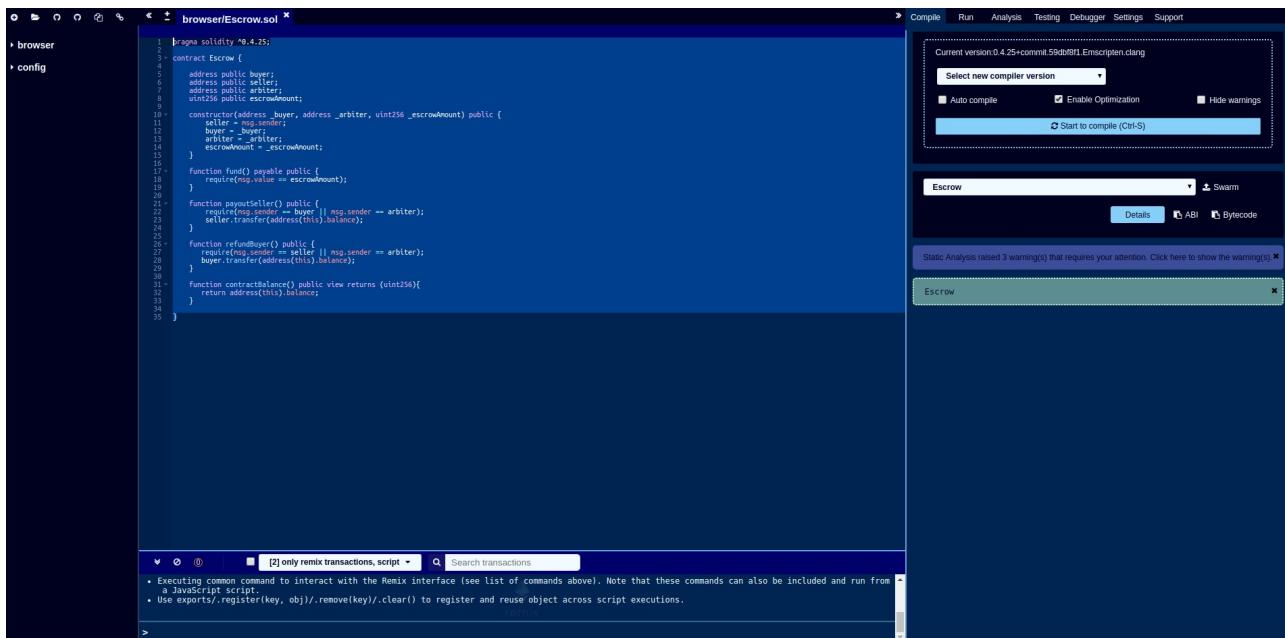
// Give a voter(s) the right to vote on this ballot.
// May only be called by the chairperson.
function giveRightToVote(address voter) public {
    Voter storage sender = voters[msg.sender];
    while (voter != delegate) {
        if (voter == msg.sender) break;
        if (voter == delegate) return;
        sender.voted = true;
        sender.proposal = 0;
        Voter storage delegate = voters[voter];
        if (proposal[delegate].voteCount < sender.weight) {
            proposal[delegate].voteCount += sender.weight;
        } else {
            delegate.weight += sender.weight;
        }
    }
}

// Delegate your vote to the voter(s).
function delegate(address to) public {
    Voter storage sender = voters[msg.sender];
    while (to != delegate) {
        if (to == msg.sender) break;
        if (to == delegate) return;
        sender.voted = true;
        sender.proposal = 0;
        Voter storage delegate = voters[to];
        if (proposal[delegate].voteCount < sender.weight) {
            proposal[delegate].voteCount += sender.weight;
        } else {
            delegate.weight += sender.weight;
        }
    }
}

// Give a single vote to proposal i<(toProposal).
function voteForProposal(uint8 toProposal) public {
    Voter storage sender = voters[msg.sender];
    if (sender.voted || toProposal >= proposals.length) return;
    sender.voted = true;
    sender.vote = toProposal;
    proposal[toProposal].voteCount += sender.weight;
}

function winningProposal() public constant returns (uint8 _winningProposal) {
    uint256 winningVoteCount;
    for (uint8 prop = 0; prop < proposals.length; prop++) {
        if (proposal[prop].voteCount > winningVoteCount) {
            winningVoteCount = proposal[prop].voteCount;
            _winningProposal = prop;
        }
    }
}
```

1. open and compile Escrow.sol



```
pragma solidity >=0.4.25;
contract Escrow {
    address public buyer;
    address public seller;
    address public arbiter;
    uint256 public escrowAmount;
    constructor(address _buyer, address _arbiter, uint256 _escrowAmount) public {
        buyer = _buyer;
        arbiter = _arbiter;
        escrowAmount = _escrowAmount;
    }

    function fund() payable public {
        require(msg.value == escrowAmount);
    }

    Function payoutSeller() public {
        require(msg.sender == buyer || msg.sender == arbiter);
        seller.transfer(address(this).balance);
    }

    function refundBuyer() public {
        require(msg.sender == buyer || msg.sender == arbiter);
        buyer.transfer(address(this).balance);
    }

    function contractBalance() public view returns (uint256) {
        return address(this).balance;
    }
}
```

2. switch to run tab and check constructor

The screenshot shows the Remix IDE interface with the following details:

- Contract Definition:** Escrow
- Environment:** JavaScript VM (VM)
- Account:** 0xca3...a733c (100 ether)
- Gas limit:** 3000000
- Value:** 0 wei

Escrow contract details:

- Deploy address: buyer.address_arbiter.uint256_escrowAmount
- Or
- All Address Load contract from Address

Transactions recorded: 0

Deployed Contracts:

Currently you have no contract instances to interact with.

Bottom Bar:

- Only remix transactions, script
- Search transactions

Contract Code (browserEscrow.sol):

```
pragma solidity ^0.4.25;

contract Escrow {
    address public buyer;
    address public seller;
    address public arbiter;
    uint256 public escrowAmount;

    constructor(address _buyer, address _arbiter, uint256 _escrowAmount) public {
        buyer = _buyer;
        seller = _seller;
        arbiter = _arbiter;
        escrowAmount = _escrowAmount;
    }

    function fund() payable public {
        require(msg.value == escrowAmount);
    }

    function payoutSeller() public {
        require(msg.sender == buyer || msg.sender == arbiter);
        seller.transfer(address(this).balance);
    }

    function refundBuyer() public {
        require(msg.sender == seller || msg.sender == arbiter);
        buyer.transfer(address(this).balance);
    }

    function contractBalance() public view returns (uint256) {
        return address(this).balance;
    }
}
```

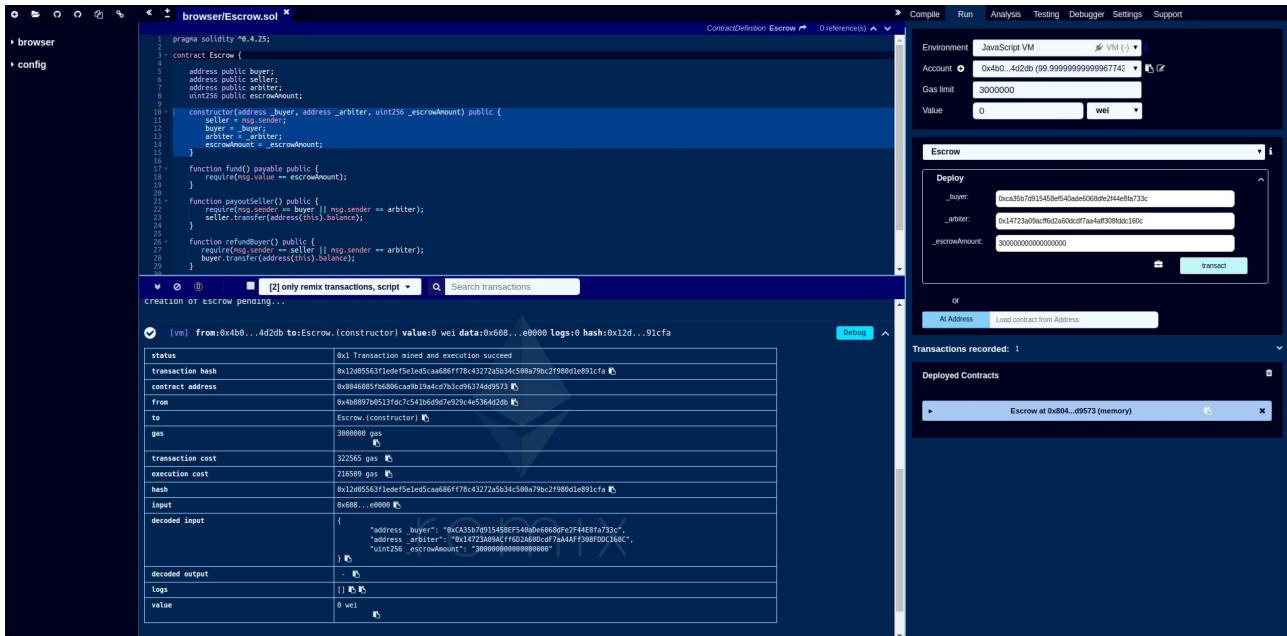
Bottom Notes:

- Executing common command to interact with the Remix interface (see list of commands above). Note that these commands can also be included and run from a JavaScript script.
- Use exports/.register(key, obj)/.remove(key)/.clear() to register and reuse object across script executions.

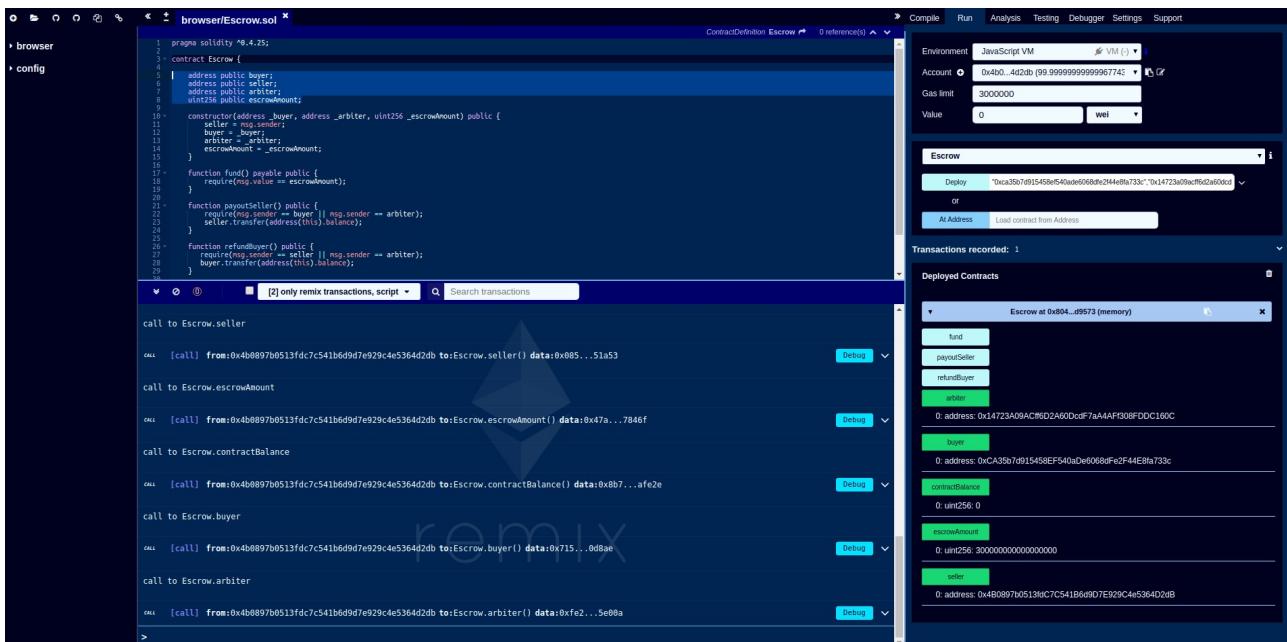
3. convert 0.3 ether to wei for escrowAmount parameter

 EtherConverter.online

4. drop down Escrow constructor form, fill out arguments and deploy (transact)



5. check some fields to see contractBalance is 0



6. switch to buyer address, set "Value" to 0.3 and select Ether, then click fund

- 0.3 Ether have been sent to the escrow contract instance
- After you click fund, the "Value" field will be reset to 0

```

pragma solidity ^0.4.25;
contract Escrow {
    address public buyer;
    address public seller;
    address public arbiter;
    uint256 public escrowAmount;
    constructor(address _buyer, address _arbiter, uint256 _escrowAmount) public {
        buyer = msg.sender;
        seller = _buyer;
        arbiter = _arbiter;
        escrowAmount = _escrowAmount;
    }
    function fund() payable public {
        require(msg.value == escrowAmount);
    }
    function payoSeller() public {
        require(msg.sender == buyer || msg.sender == arbiter);
        seller.transfer(address(this).balance);
    }
    function refundBuyer() public {
        require(msg.sender == seller || msg.sender == arbiter);
        buyer.transfer(address(this).balance);
    }
}

```

7. click contractBalance to see it has changed from 0 to 3000000000000000000

```

pragma solidity ^0.4.25;
contract Escrow {
    address public buyer;
    address public seller;
    address public arbiter;
    uint256 public escrowAmount;
    constructor(address _buyer, address _arbiter, uint256 _escrowAmount) public {
        buyer = msg.sender;
        seller = _buyer;
        arbiter = _arbiter;
        escrowAmount = _escrowAmount;
    }
    function fund() payable public {
        require(msg.value == escrowAmount);
    }
    function payoSeller() public {
        require(msg.sender == buyer || msg.sender == arbiter);
        seller.transfer(address(this).balance);
    }
    function refundBuyer() public {
        require(msg.sender == seller || msg.sender == arbiter);
        buyer.transfer(address(this).balance);
    }
}

```

8. try to refundBuyer and see it fail

- Because the contract defines that you as the buyer cant refund

```

pragma solidity >=0.4.25;
contract Escrow {
    address public buyer;
    address public seller;
    address public arbiter;
    uint256 public escrowAmount;
    constructor(address _buyer, address _arbiter, uint256 _escrowAmount) public {
        buyer = msg.sender;
        seller = _buyer;
        arbiter = _arbiter;
        escrowAmount = _escrowAmount;
    }
    function fund() payable public {
        require(msg.sender == buyer || msg.sender == arbiter);
    }
    function payoutSeller() public {
        require(msg.sender == seller || msg.sender == arbiter);
        buyer.transfer(address(this).balance);
    }
    function refundBuyer() public {
        require(msg.sender == seller || msg.sender == arbiter);
        buyer.transfer(address(this).balance);
    }
}

```

9. switch to seller address and try to payoutSeller and see it fail

- Because the contract defines that you as a seller cant payout

```

[2] only remix transactions, script < Search transactions
hash 0x1431465f31910faabaa15e6bb8bf11881356fad99fe6ac0ff3c58e795687c5
Input 0x66...04288
decoded input () 
decoded output () 
Logs () 
value 3000000000000000000 wei

call to Escrow.contractBalance
out [call] from:0xca35b7d915458ef540ade6068dfc2f44e@fa733c to:Escrow.contractBalance() data:0x8b7...afe2e
transact to Escrow.refundBuyer pending ...
[vm] from:0xa3...a733c to:Escrow.refundBuyer() 0x804...d9573 value:0 wei data:0xe8a...61c8 logs:0 hash:0x479...79b07
transact to Escrow.refundBuyer errored: VM error: revert.
revert The transaction has been reverted to the initial state.
Note: The constructor should be payable if you send value. Debug the transaction to get more information.
>

```

10. switch to arbiter address and decide to cancel by refundBuyer

The screenshot shows the Remix IDE interface with the Escrow contract code in the left panel. The right panel displays the environment settings (JavaScript VM, Account 0x147...c160c, Gas limit 3000000, Value 0 ether), a transaction history section titled "Transactions recorded" (with one entry for refundBuyer), and a "Deployed Contracts" section showing the deployed Escrow contract at address 0x804...d9573.

```

pragma solidity ^0.4.25;
contract Escrow {
    address public buyer;
    address public seller;
    address public arbiter;
    uint256 public escrowAmount;
    constructor(address _buyer, address _arbiter, uint256 _escrowAmount) public {
        buyer = _buyer;
        arbiter = _arbiter;
        escrowAmount = _escrowAmount;
    }
    function fund() payable public {
        require(msg.value == escrowAmount);
    }
    function payoutSeller() public {
        require(msg.sender == buyer || msg.sender == arbiter);
        seller.transfer(address(this).balance);
    }
    function refundBuyer() public {
        require(msg.sender == seller || msg.sender == arbiter);
        buyer.transfer(address(this).balance);
    }
}

```

11. check escrowAmount and see it has changed from 3000000000000000000000000 to 0

This screenshot is similar to the previous one but shows a different transaction in the "Transactions recorded" section. The transaction details for refundBuyer() show the gas used and the value sent (0 wei).

Call to Escrow.contractBalance

```

call [call] from:0x14723a09acff6d2a60cd7aa4aff308fddc160c to:Escrow.contractBalance() data:0xb87...afe2e

```

12. Go to "Transactions recorded" and click the dropdown and "save icon"

The screenshot shows the Remix IDE interface. On the left, there's a file tree with files like browser/Escrow.sol, browser/escrow.sol, ballot.sol, ballot_test.sol, and config. The main area displays the Solidity code for the Escrow contract. The code defines a constructor, a payable fund() function, a payable payoutSeller() function, a payable refundBuyer() function, and a view contractBalance() function. It uses require statements to check msg.sender against buyer, arbiter, and seller addresses. It also uses transfer functions to move ether between these addresses. The right side of the interface shows deployment details: Environment (JavaScript VM), Account (0x147...c180c (99.9999999999996968)), Gas limit (3000000), Value (0 ether). Below this is the 'Escrow' section with a Deploy button and an At Address input field. A Deployed Contracts section shows 'Escrow at 0x804...d9573 (memory)'. At the bottom, there's a search bar for transactions.

```
pragma solidity ^0.4.25;
contract Escrow {
    address public buyer;
    address public seller;
    address public arbiter;
    uint256 public escrowAmount;
    constructor(address _buyer, address _arbiter, uint256 _escrowAmount) public {
        buyer = _buyer;
        arbiter = _arbiter;
        escrowAmount = _escrowAmount;
    }
    function fund() payable public {
        require(msg.value == escrowAmount);
    }
    function payoutSeller() public {
        require(msg.sender == buyer || msg.sender == arbiter);
        seller.transfer(address(this).balance);
    }
    function refundBuyer() public {
        require(msg.sender == seller || msg.sender == arbiter);
        buyer.transfer(address(this).balance);
    }
    function contractBalance() public view returns (uint256){
        return address(this).balance;
    }
}
```

13. choose file name and click ok

This screenshot shows the same Remix IDE setup as the previous one, but with a modal dialog box in the center. The dialog says 'Transactions will be saved in a file under browser' and 'scenario.json'. It has 'OK' and 'Cancel' buttons. The rest of the interface is identical to the first screenshot, showing the Solidity code and deployment details.

14. see the saved recording json file

The screenshot shows the Remix IDE interface. On the left, there are two tabs: "browser/Escrow.sol" and "browser/scenario.json". The "browser/Escrow.sol" tab displays the Solidity code for the Escrow contract, which includes functions like `fund`, `payeeSeller`, `refundBuyer`, `arbitrator`, `buyer`, `contractBalance`, `getBalance`, and `isEscrowed`. The "browser/scenario.json" tab shows a JSON configuration for testing, including accounts, transactions, bytecode, and constructor parameters. On the right side, there's a "Deploy" section with fields for account (0x147...), gas limit (3000000), and value (0 wei). Below it, a "Transactions recorded" section lists a single transaction hash. A "Deployed Contracts" section shows the deployed contract at address 0xd0d...bcf7.

15. for example change addresses and constructor params and click PLAY to replay

- see all 5 transactions repeat with the changed data
- see another instance of the newly deployed contract

This screenshot shows the same Remix IDE setup as the previous one, but with changes made to the "scenario.json" file. The "accounts" section now includes a new account (0x8c583b76915458e5f540ae6068df2f44e0fa733c) and a different address for the buyer (0x1473a099acff6d2a68dc0f7aa4af38bf0dc16bc). The "constructor" parameters have also been modified. The "Deploy" section still shows the original account (0x147...). The "Transactions recorded" section now lists five transactions, each with a different hash due to the changes in the constructor parameters. The "Deployed Contracts" section shows two instances of the Escrow contract: one at 0xd0d...bcf7 and another at 0x692...77b3a.

16. check all read only methods and see values have the changed ones

- contractBalance is 0 because everything has already been refunded to the buyer by arbiter

The screenshot shows the Remix IDE interface with the following details:

- Contract Definition:** Escrow
- Gas limit:** 3000000
- Value:** 0 Wei
- Deploy:** 0xa... (Deployment address)
- At Address:** Load contract from Address
- Transactions recorded:** 0
- Deployed Contracts:**
 - Escrow at 0x692... (memory)
 - Escrow at 0xd0... (memory)
- Transactions:**
 - [vm] from:0xbdd...92148 to:Escrow.refundBuyer() 0xd... value:0 wei data:0xe0...61cc logs:0 hash:0x3f...d2719
 - call to Escrow.seller
 - [call] from:0x14723a...90cff6d2a0dcdff7aa4aff308ffddc160c to:Escrow.seller() data:0x05...51a53
 - call to Escrow.escrowAmount
 - [call] from:0x14723a...90cff6d2a0dcdff7aa4aff308ffddc160c to:Escrow.escrowAmount() data:0x47...7846f
 - call to Escrow.contractBalance
 - [call] from:0x14723a...90cff6d2a0dcdff7aa4aff308ffddc160c to:Escrow.contractBalance() data:0xb7...afe2e
 - call to Escrow.buyer
 - [call] from:0x14723a...90cff6d2a0dcdff7aa4aff308ffddc160c to:Escrow.buyer() data:0x71...040ea
 - call to Escrow.arbitrator
 - [call] from:0x14723a...90cff6d2a0dcdff7aa4aff308ffddc160c to:Escrow.arbitrator() data:0xfe2...5fe0a