0001 - 概览

一旦拍卖结束，所有的投标人公告出价，胜出者就会出现。接下来发生什么？卖方必须将货物发送给买方并获得资金。现在有几件事情可能会出错。

- 您如何信任卖家遵循并发送产品？卖方可以很容易地拿走钱，消失。
- 如果卖方确实交付了产品，但买方不承认他们收到了该产品呢?
- 如果产品损坏，买方应退还，但卖方拒绝这样做?
- 为了解决所有这些问题，我们创建了一个`Multisig Escrow`合同，以买方，卖方和第三方仲裁者作为参与者，来管理金额。如果3位参与者中的至少2位同意这样做，则托管金额只能发放给卖方或退还给买方。

在接下来的几节中，我们将实现Escrow合同，并添加在拍卖结束时在运行时创建Ethereum Escrow合同的功能。

0002 - 托管合约 Contract

Escrow.sol

```
pragma solidity ^0.4.13;
contract Escrow {
 uint public productId;
 address public buyer;
 address public seller;
 address public arbiter;
 uint public amount;
 bool public fundsDisbursed;
 mapping (address => bool) releaseAmount;
 uint public releaseCount;
 mapping (address => bool) refundAmount;
 uint public refundCount;

 event CreateEscrow(uint _productId, address _buyer, address _seller, address _arbiter);
 event UnlockAmount(uint _productId, string _operation, address _operator);
 event DisburseAmount(uint _productId, uint _amount, address _beneficiary);

 function Escrow(uint _productId, address _buyer, address _seller, address _arbiter) payable public {
  productId = _productId;
  buyer = _buyer;
  seller = _seller;
  arbiter = _arbiter;
  amount = msg.value;
  fundsDisbursed = false;
  CreateEscrow(_productId, _buyer, _seller, _arbiter);
 }

 function escrowInfo() view public returns (address, address, address, bool, uint, uint) {
  return (buyer, seller, arbiter, fundsDisbursed, releaseCount, refundCount);
 }

 function releaseAmountToSeller(address caller) public {
  require(!fundsDisbursed);
  if ((caller == buyer || caller == seller || caller == arbiter) && releaseAmount[caller] != true) {
   releaseAmount[caller] = true;
   releaseCount += 1;
   UnlockAmount(productId, "release", caller);
  }

  if (releaseCount == 2) {
   seller.transfer(amount);
   fundsDisbursed = true;
   DisburseAmount(productId, amount, seller);
  }
 }

 function refundAmountToBuyer(address caller) public {
  require(!fundsDisbursed);
  if ((caller == buyer || caller == seller || caller == arbiter) && refundAmount[caller] != true) {
   refundAmount[caller] = true;
   refundCount += 1;
   UnlockAmount(productId, "refund", caller);
  }

  if (refundCount == 2) {
   buyer.transfer(amount);
   fundsDisbursed = true;
   DisburseAmount(productId, amount, buyer);
  }
 }
}
```

0003 - 增加仲裁

product.html

```
<form id="finalize-auction" class="col-sm-6">
<input type="hidden" name="product-id" id="product-id" />
<button type="submit" class="btn form-submit">Finalize Auction
</form>
```

EcommerceStore.sol

```
import "contracts/Escrow.sol";
contract EcommerceStore {
......
 mapping (uint => address) productEscrow;
......
......
}
function finalizeAuction(uint _productId) public {
 Product memory product = stores[productIdInStore[_productId]][_productId];

  require(now > product.auctionEndTime);
 require(product.status == ProductStatus.Open);
 require(product.highestBidder != msg.sender);
 require(productIdInStore[_productId] != msg.sender);

 if (product.totalBids == 0) {
  product.status = ProductStatus.Unsold;
 } else {

  Escrow escrow = (new Escrow).value(product.secondHighestBid)(_productId, product.highestBidder, productIdInStore[_productId], msg.sender);
  productEscrow[_productId] = address(escrow);
  product.status = ProductStatus.Sold;

  uint refund = product.highestBid - product.secondHighestBid;
  product.highestBidder.transfer(refund);

 }
 }
 function escrowAddressForProduct(uint _productId) view public returns (address) {
 return productEscrow[_productId];
 }

 function escrowInfo(uint _productId) view public returns (address, address, address, bool, uint, uint) {
 return Escrow(productEscrow[_productId]).escrowInfo();
}
```

app.js

```
$("#finalize-auction").submit(function(event) {
  $("#msg").hide();
  let productId = $("#product-id").val();
  EcommerceStore.deployed().then(function(i) {
  i.finalizeAuction(parseInt(productId), {from: web3.eth.accounts[0]}).then(
   function(f) {
   $("#msg").show();
   $("#msg").html("The auction has been finalized and winner declared.");
   console.log(f)
   location.reload();
   }
  ).catch(function(e) {
   console.log(e);
   $("#msg").show();
   $("#msg").html("The auction can not be finalized by the buyer or seller, only a third party aribiter can finalize it");
  })
  });
  event.preventDefault();
});
```

更新renderProductDetails函数

```
function renderProductDetails(productId) {
 EcommerceStore.deployed().then(function(i) {
 i.getProduct.call(productId).then(function(p) {
  console.log(p);
  let content = "";
  ipfs.cat(p[4]).then(function(stream) {
        console.log(stream);
        let content = Utf8ArrayToStr(stream);
        $("#product-desc").append("<div>" + content + "</div>");

});

  $("#product-image").append("<img src='https://ipfs.io/ipfs/" + p[3] + "' width='250px' />");
  $("#product-price").html(displayPrice(p[7]));
  $("#product-name").html(p[1].name);
  $("#product-auction-end").html(displayEndHours(p[6]));
  $("#product-id").val(p[0]);
  $("#revealing, #bidding, #finalize-auction, #escrow-info").hide();
  let currentTime = getCurrentTimeInSeconds();
   if (parseInt(p[8]) == 1) {
  $("#product-status").html("Product sold");
  } else if(parseInt(p[8]) == 2) {
  $("#product-status").html("Product was not sold");
  } else if(currentTime < parseInt(p[6])) {
  $("#bidding").show();
  } else if (currentTime < (parseInt(p[6]) + 600)) {
  $("#revealing").show();
  } else {
  $("#finalize-auction").show();
  }
 })
 })
}
```

0004 - 付款或者退款

EcommerceStore.sol

```
function releaseAmountToSeller(uint _productId) public {
  Escrow(productEscrow[_productId]).releaseAmountToSeller(msg.sender);
}

function refundAmountToBuyer(uint _productId) public {
  Escrow(productEscrow[_productId]).refundAmountToBuyer(msg.sender);
}
```

app.js

如果产品信息为 `Sold`，显示托管信息，替换 `$("#product-status").html("Product sold");` 代码显示托管结果。

```
if (parseInt(p[8]) == 1) {
   EcommerceStore.deployed().then(function(i) {
    $("#escrow-info").show();
    i.highestBidderInfo.call(productId).then(function(f) {
     if (f[2].toLocaleString() == '0') {
      $("#product-status").html("Auction has ended. No bids were revealed");
     } else {
      $("#product-status").html("Auction has ended. Product sold to " + f[0]
 + " for " + displayPrice(f[2]) +
       "The money is in the escrow. Two of the three participants (Buyer, Se
ller and Arbiter) have to " +
       "either release the funds to seller or refund the money to the buyer"
);
     }
    })
    i.escrowInfo.call(productId).then(function(f) {
     $("#buyer").html('Buyer: ' + f[0]);
     $("#seller").html('Seller: ' + f[1]);
     $("#arbiter").html('Arbiter: ' + f[2]);
     if(f[3] == true) {
      $("#release-count").html("Amount from the escrow has been released");
     } else {
      $("#release-count").html(f[4] + " of 3 participants have agreed to rel
ease funds");
      $("#refund-count").html(f[5] + " of 3 participants have agreed to refund the buyer");
     }
    })
   })
  }
```

添加付款和退款

```
$("#release-funds").click(function() {
   let productId = new URLSearchParams(window.location.search).get('id');
   EcommerceStore.deployed().then(function(f) {
    $("#msg").html("Your transaction has been submitted. Please wait for few seconds for the confirmation").show();
    console.log(productId);
    f.releaseAmountToSeller(productId, {from: web3.eth.accounts[0], gas: 440000}).then(function(f) {
     console.log(f);
     location.reload();
    }).catch(function(e) {
     console.log(e);
    })
   });
  });

  $("#refund-funds").click(function() {
   let productId = new URLSearchParams(window.location.search).get('id');
   EcommerceStore.deployed().then(function(f) {
    $("#msg").html("Your transaction has been submitted. Please wait for few seconds for the confirmation").show();
    f.refundAmountToBuyer(productId, {from: web3.eth.accounts[0], gas: 440000}).then(function(f) {
     console.log(f);
     location.reload();
    }).catch(function(e) {
     console.log(e);
    })
   });

   alert("refund the funds!");
  });
```