IMPLEMENTATION:

1)SINGLE TRANSPORTER POD:

Pre-Condition before initiating the contract:
This function sets the pre-condition for a physical item that is item price, delivery duration and buyer verification time window and terms and condition IPFShash (which can be viewed by each agent involved in delivery process.

```
string memory itemID) public payable {
    seller = _seller;
    buyer = _buyer;
    transporter = _transporter;
    arbitrator = _arbitrator;
    attestaionAuthority = _attestationAuthority;

    itemPrice = 0.001 ether;
    itemID = itemID;
    deliveryDuration = 2 hours; // 2 hours
    buyerVerificationTimeWindow = 2 minutes; // Time for the buyer to verify keys after transporter entered the keys
    TermsIPFS_Hash = "QmWWQSuPMS6aXCbZKpEjPHPUZN2NjB3YrhJTHsV4X3vb2td";

    state = contractState.waitingForVerificationbySeller;
}
```

Signed Terms and Conditions Function:
This function collects the collateral from each agent and change the state of contract to indicate weather double deposit is done or not.

- Collateral (Double Deposit) is being deposited by each agent: seller, buyer and transporter.
- State verified before depositing any collateral.
- Once state verified and collateral deposited by buyer, seller, transporter cancellable flag set to true.

```
function SignTermsAndConditions() public payable costs OnlySeller_Buyer_Transporter {
    if(msg.sender == seller) {
        require(state == contractState.waitingForVerificationbySeller);
        emit TermsAndConditionsSignedBy("Terms and Conditiond verified : ", msg.sender);
        emit collateralWithdrawnSuccessfully("Double deposit is withdrawn successfully from: ", msg.sender);
        state = contractState.waitingForVerificationbyTransporter;
    }
    else if(msg.sender == transporter) {
        require(state == contractState.waitingForVerificationbyTransporter);
        emit TermsAndConditionsSignedBy("Terms and Conditiond verified : ", msg.sender);
        emit collateralWithdrawnSuccessfully("Double deposit is withdrawn successfully from: ", msg.sender);
        state = contractState.waitingForVerificationbyBuyer;
    }
    else if(msg.sender == buyer) {
        require(state == contractState.waitingForVerificationbyBuyer);
        emit TermsAndConditionsSignedBy("Terms and Conditiond verified : ", msg.sender);
        emit collateralWithdrawnSuccessfully("Double deposit is withdrawn successfully from: ", msg.sender);
        state = contractState.MoneyWithdrawn;
        cancellable[seller] = true;
        cancellable[buyer] = true;
        cancellable[transporter] = true;
    }
}
```

Create Package Key Function:
This function creates a package key for a transporter to carry out the package.

- Contract state verified
- Now package can't be cancelled so cancellable transporter flag set to false.
- Generate a key hash and hand it over to transporter.

```
function createPackageAndKey() public payable OnlySeller {
    require(state == contractState.MoneyWithdrawn);

    cancellable[msg.sender] = false;
    cancellable[transporter]=false;
    keyTr = uint(keccak256(
        abi.encodePacked(itemID, transporter, block.timestamp)
    ))/100000000000000000000000000000000000000000000000000000000000000000000000;
    state = contractState.PackageAndTransporterKeyCreated;
    emit PackageCreatedBySeller("Package created and Key given to transporter by the sender ", msg.sender);
}
```

Delivery Package Function:
This function verifies weather transporter received key or not and changed the state of contract to item_on_way.

- Contract state of package_transporter_key_created verified.
- Saves the delivery time.
- Now buyer can't cancel the order anymore that's why cancellable flag for buyer is not null.
- Contract state changed to item on the way.

```
function deliverPackage() public OnlyTransporter {
    require(state == contractState.PackageAndTransporterKeyCreated);
    startdeliveryBlocktime = block.timestamp;//save the delivery time
    cancellable[buyer] = false;
    emit PackageIsOnTheWay("The package is being delivered and the key is received by the ", msg.sender);
    state = contractState.ItemOnTheWay;
}
```

Request Package Key Function:
This function generates key for buyer and verifies weather package is received by buyer or not.

- Contract state verified.
- Buyer requests the key to verify the transporter.
- Contract state changed to package_key_given_to_buyer.

```
function requestPackageKey() public OnlyBuyer {
    require(state == contractState.ItemOnTheWay);
    keyBr = uint(keccak256(
        abi.encodePacked(itemID, buyer, block.timestamp)
    ))/100000000000000000000000000000000000000000000000000000000000000000000000;
    state = contractState.PackageKeyGivenToBuyer;
    emit PackageKeyGivenToBuyer("The package Key is given to the ", msg.sender);
}
```

Verify Transporter Function:
This function verifies package state and transporter enters the key which in turn invokes internal verification function.

- Contract state verified as package_key_given_to_buyer.
- Transporter entered keys and internal verification function invoked to check key entered by transporter is correct or not.
- Contract state changed to Arrived_to_destination
- Start the time window for buyer to enter keys

```
function verifyTransporter(string memory keyT, string memory keyR) public OnlyTransporter {
    require(state == contractState.PackageKeyGivenToBuyer);
    emit ArrivedToDestination("Transporter Arrived To Destination and entered keys " , msg.sender);
    verificationHash[transporter] = keccak256(
            abi.encodePacked(keyT, keyR)
        );
    state = contractState.ArrivedToDestination;
    startEntryTransporterKeysBlocktime = block.timestamp;
}
```

Verify Buyer Function:
This function verifies package state and buyer enters the key which in turn invokes internal verification function.

- Contract state verified as Arrived_to_destination
- Buyer entered keys and waiting for settlement.
- Contract state changed to buyer_key_entered.
- Internal Verification function invoked.

```
function verifyKeyBuyer(string memory keyT, string memory keyR) public OnlyBuyer {
    require(state == contractState.ArrivedToDestination);
    emit BuyerEnteredVerificationKeys("Reciever entered keys, waiting for payment settlement", msg.sender);
    verificationHash[buyer] = keccak256(
            abi.encodePacked(keyT, keyR)
        );
    state = contractState.buyerKeysEntered;
    verification();
}
```

Internal Verification Function:
This verification function ensures that weather the keys entered by both buyer and seller are correct or there is dispute.

- Contract state verified as buyer_key_entered.
- Values of verification hash of buyer and transporter must be equal.
- Settle prices for each agent.
- In case values of verification hash is not equal balance of contract is transferred to arbitrator.
- After Successful verification contract state changed to payement_settle_success otherwise contract state changed to aborted.

```
function verification() internal {
    require(state == contractState.buyerKeysEntered);
    if(verificationHash[transporter] == verificationHash[buyer]){
        emit SuccessfulVerification("Payment will shortly be settled , successful verification!");
        buyer.transfer(itemPrice);
        transporter.transfer((2*itemPrice) + ((10*itemPrice)/100));//receiver gets 10% of item price delivered
        seller.transfer((2*itemPrice)+((90*itemPrice)/100));
        state = contractState.PaymentSettledSuccess;
    }
    else {
        //trusted entity the Arbitrator resolves the issue
        emit VerificationFailure("Verification failed , keys do not match. Please solve the dispute off chain. No refunds.");
        state = contractState.DisputeVerificationFailure;
        arbitrator.transfer(address(this).balance);//all ether with the contract
        state = contractState.EtherWithArbitrator;
        emit EtherTransferredToArbitrator("Due to dispute all Ether deposits have been transferred to arbitrator ", arbitrator);
        state = contractState.Aborted;
    }
}
```

Cancel Transaction Function:
This function cancels the transaction and refunds the item_price to each agent like seller, buyer and transporter.

- Contract state changed to cancellation_refund
- Price of each item is refunded to each agent
- Printed out reason for refund and changed the contract state to aborted.

```
function cancelTransaction(string memory reason)public OnlySeller_Buyer_Transporter{
 require(cancellable[msg.sender] == true);
 state = contractState.CancellationRefund;
 //everyone gets a refund
 seller.transfer(2*itemPrice);
 buyer.transfer(2*itemPrice);
 transporter.transfer(2*itemPrice);
 emit CancellationRequest(msg.sender, " has requested a cancellation due to: ", reason );
 state = contractState.Aborted;
 }
```

Return_Key Function:
This function helps to printout the transporter and buyer key hashes.

- For seller and transporter keys must be same.

```
function returnKey() public view returns(uint){
    if(msg.sender == transporter || msg.sender == seller){
        return keyTr;
    }
    else if(msg.sender == buyer){
        return keyBr;
    }
}
```

IsRefundable Function:

- This function ensures the start_delivery and delivery duration meet the pre-condition set in the start of contract.

IsBuyerExceededTime Function:

- This function ensures weather the transporter exceeded the time of delivery and buyer key verification time.

```
function isRefundable() public view OnlyBuyer returns(bool){
  return (block.timestamp > startdeliveryBlocktime+deliveryDuration &&
  (state == contractState.ItemOnTheWay || state == contractState.PackageKeyGivenToBuyer));
}
function isBuyerExceededTime() public view OnlyTransporter returns(bool){
  return (block.timestamp > startEntryTransporterKeysBlocktime + buyerVerificationTimeWindow &&
  state == contractState.ArrivedToDestination);
}
```

Successful Payment Settlement Verification:

```
0  listen on network   Search with transaction hash or address

[vm] from:0xf1c...e512d to:POD.(constructor) value:0 wei data:0x608...00000 logs:0 hash:0x24a...36fbc   Debug

transact to POD.SignTermsAndConditions pending ...

[vm] from:0x1af...d869e to:POD.SignTermsAndConditions() 0xb4d...67903 value:2000000000000000 wei data:0x577...da7d2 logs:2
hash:0x525...bcb6b   Debug

transact to POD.SignTermsAndConditions pending ...

[vm] from:0xf1c...e512d to:POD.SignTermsAndConditions() 0xb4d...67903 value:2000000000000000 wei data:0x577...da7d2 logs:2
hash:0xe4e...d40a8   Debug

transact to POD.SignTermsAndConditions pending ...

[vm] from:0x558...e1255 to:POD.SignTermsAndConditions() 0xb4d...67903 value:2000000000000000 wei data:0x577...da7d2 logs:2
hash:0xa4e...c4330   Debug

transact to POD.createPackageAndKey pending ...
```

```
0  listen on network   Search with transaction hash or address

[vm] from:0x1af...d869e to:POD.createPackageAndKey() 0xb4d...67903 value:0 wei data:0x3f8...9301e logs:1 hash:0x742...93349   Debug

transact to POD.deliverPackage pending ...
```

```
[vm] from:0xf1c...e512d to:POD.deliverPackage() 0xb4d...67903 value:0 wei data:0xb1a...25c3e logs:1 hash:0x627...d572e   Debug

transact to POD.requestPackageKey pending ...

[vm] from:0x558...e1255 to:POD.requestPackageKey() 0xb4d...67903 value:0 wei data:0xaef...d54a8 logs:1 hash:0x8b5...7f27a   Debug

transact to POD.verifyKeyBuyer pending ...
```

```
[vm] from:0xf1c...e512d to:POD.verifyTransporter(string,string) 0xb4d...67903 value:0 wei data:0x41f...00000 logs:1
hash:0xb8a...a1655

transact to POD.verifyKeyBuyer pending ...

[vm] from:0x558...e1255 to:POD.verifyKeyBuyer(string,string) 0xb4d...67903 value:0 wei data:0x5c2...00000 logs:2
hash:0x4bd...36ac6

call to POD.state

CALL [call] from:0x558ae1b80056303FA92B70966bCA3e138CCE1255 to:POD.state() data:0xc19...d93fb

>
```

Dispute Handling:

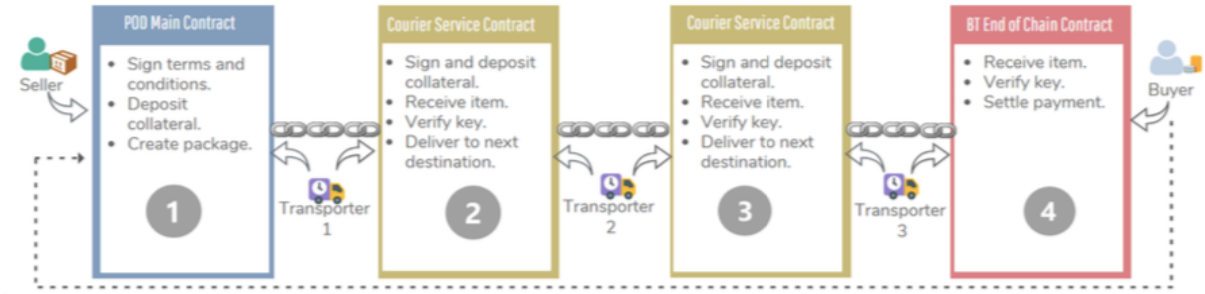
- If any of the key entered wrong either by buyer or transporter the smart contract amount directly goes to arbitrator.
- Limited amount of time is given to buyer to enter keys once transporter reaches and entered keys into the system to test that I set time window to 2 minutes if the buyer exceeded time then dispute goes to arbitrator.

- Similarly, if the transporter exceeded the time window of delivery which we set to 2 hours then also the dispute occurs, and arbitrator resolves issues.

```
0  listen on network   Search with transaction hash or address

decoded input
{
        "string keyT": "0x8b749ab263348cc6c3e4e06afe6b07dfb79c13e18049fec9b0181e3a0133c5a4",
        "string keyR": "0x8c749ab263348cc6c3e4e06afe6b07dfb79c13e18049fec9b0181e3a0133c5a4"
} 

decoded output
{} 

logs
[
        {
                "from": "0xb7b757aed1de0f601548fc4f27d970e68cdccded",
                "topic": "0xd58c2147902b2eb7ba2e7297446aad01cc528c23b8b37c0bf79b1a8dda675f80",
                "event": "BuyerEnteredVerificationKeys",
                "args": {
                        "0": "Reciever entered keys, waiting for payment settlement",
                        "1": "0x9A567E3df8912fA2E44325da1D0046Cfeb6B1475",
                        "info": "Reciever entered keys, waiting for payment settlement",
                        "entityAddress": "0x9A567E3df8912fA2E44325da1D0046Cfeb6B1475",
                        "length": 2
                }
        },
        {
                "from": "0xb7b757aed1de0f601548fc4f27d970e68cdccded",
                "topic": "0x4ed0570313882c4b1660d5a3a0fdca6fbeddba1bfca75a125dcd7e403f192442",
                "event": "VerificationFailure",
                "args": {
                        "0": "Verification failed , keys do not match. Please solve the dispute
off chain. No refunds.",
                        "info": "Verification failed , keys do not match. Please solve the dispute
off chain. No refunds.",
                        "length": 1
                }
        },
        {
                "from": "0xb7b757aed1de0f601548fc4f27d970e68cdccded",
                "topic": "0xfc0cf60f1f5c19e05c32dfb2dff6c9823cdc87846e26a2e2335a01cb184ab2f3",
                "event": "EtherTransferredToArbitrator",
                "args": {
                        "0": "Due to dispute all Ether deposits have been transferred to
arbitrator ",
                        "1": "0x98b11807E2f391217a8Af9F51f8eA7f63981CE0d",
                        "info": "Due to dispute all Ether deposits have been transferred to
arbitrator ",
                        "entityAddress": "0x98b11807E2f391217a8Af9F51f8eA7f63981CE0d",
                        "length": 2
                }
        }
]
```

2)MULTIPLE TRANSPORTERS POD:

The flow is divided into 3 different POD smart-contracts: **POD Main Contract**, **Courier Contract** (transporters interaction contract) and **BT Contract** (buyer and transporter end-point contract).



Pre-Condition before initiating the contract:
This function sets the pre-condition for a physical item that is item price, delivery duration and buyer verification time window and terms and condition IPFShash (which can be viewed by each agent involved in delivery process.

```
constructor(address payable _seller,
address payable _buyer,
address payable _transporter,
address payable _arbitrator,
address payable _attestationAuthority,
address payable _courier_contract_address,
string memory _itemID) public payable {
    seller = _seller;
    buyer = _buyer;
    transporter = _transporter;
    arbitrator = _arbitrator;
    attestaionAuthority = _attestationAuthority;

    itemPrice = 1 ether;
    itemID = _itemID;
    deliveryDuration = 2 hours; // 2 hours
    buyerVerificationTimeWindow = 2 minutes; // Time for the buyer to verify keys after transporter entered the keys
    courier_contract_address=_courier_contract_address;
    TermsIPFS_Hash = "QmWWQSuPMS6aXCbZKpEjPHPUZN2NjB3YrhJTHsV4X3vb2td";

    state = contractState.waitingForVerificationbySeller;
}
```

Signed Terms and Conditions Function:
This function collects the collateral from each agent and change the state of contract to indicate weather double deposit is done or not.

- Collateral (Double Deposit) is being deposited by each agent: seller, buyer and transporter.
- State verified before depositing any collateral.
- Once state verified and collateral deposited by buyer, seller, transporter cancellable flag set to true.

```
function SignTermsAndConditions() public payable costs OnlySeller_Buyer_Transporter {
    if(msg.sender == seller) {
        require(state == contractState.waitingForVerificationbySeller);
        emit TermsAndConditionsSignedBy("Terms and Conditiond verified : ", msg.sender);
        emit collateralWithdrawnSuccessfully("Double deposit is withdrawn successfully from: ", msg.sender);
        state = contractState.waitingForVerificationbyTransporter;
    }
    else if(msg.sender == transporter) {
        require(state == contractState.waitingForVerificationbyTransporter);
        emit TermsAndConditionsSignedBy("Terms and Conditiond verified : ", msg.sender);
        emit collateralWithdrawnSuccessfully("Double deposit is withdrawn successfully from: ", msg.sender);
        state = contractState.waitingForVerificationbyBuyer;
    }
    else if(msg.sender == buyer) {
        require(state == contractState.waitingForVerificationbyBuyer);
        emit TermsAndConditionsSignedBy("Terms and Conditiond verified : ", msg.sender);
        emit collateralWithdrawnSuccessfully("Double deposit is withdrawn successfully from: ", msg.sender);
        state = contractState.MoneyWithdrawn;
        cancellable[seller] = true;
        cancellable[buyer] = true;
        cancellable[transporter] = true;
    }
}
```

CreatePackageAndKey Function:
This function creates a package key for a Transporter Number 1 to carry out the package.

- Contract state verified
- Now package can't be cancelled so cancellable transporter flag set to false.
- Generate a key hash and hand it over to transporter.

```
    function createPackageAndKey() public payable OnlySeller {
        require(state == contractState.MoneyWithdrawn);

        cancellable[msg.sender] = false;
        cancellable[transporter]=false;
        keyTr = uint(keccak256(
            abi.encodePacked(itemID, transporter, block.timestamp)
        ))/100000000000000000000000000000000000000000000000000000000000000000000000;
        state = contractState.PackageAndTransporterKeyCreated;
        emit PackageCreatedBySeller("Package created and Key given to transporter by the sender ", msg.sender);
}
```

POD Main Contract then calls Courier Contract where in Transporter passes package and its key to the next transporter.

Signed Terms and Conditions Function (Courier Contract):
This function ensures that the next transporter deposits the collateral and signs the terms and conditions.

- Collateral (Double Deposit) is being deposited by next transporter.
- State verified before depositing any collateral.
- Once state verified the contract state changed to MoneyWithdrawn.

```
function SignTermsAndConditions() public payable costs OnlyNextTransporter {

    if(msg.sender == transporter2) {
        require(state == contractState.waitingForVerificationbyTransporter);
        emit TermsAndConditionsSignedBy("Terms and Conditiond verified : ", msg.sender);
        emit collateralWithdrawnSuccessfully("Double deposit is withdrawn successfully from: ", msg.sender);
        state = contractState.MoneyWithdrawn;
    }

}
```

ConfirmedPackageReceived (Courier Contract):
This function confirms the package received by next transporter.

- Check the contract state.
- Saves the delivery time of first transporter.
- Handover the key to next transporter and changed the state to PacakageRecievedbyTransporter.

```
function ConfirmPackageReceived() public OnlyNextTransporter {
    require(state == contractState.MoneyWithdrawn);
    startdeliveryBlocktime = block.timestamp;//save the delivery time
    emit PackageKeyGivenToNextTransporter("The package is being delivered and the key is received by the Transporter", msg.sender);
    state = contractState.PackageReceivedByTansporter;
}
```

KeysEnteredByTransporter (Courier Contract):
This function confirms arrival of former transporter.

- Check the contract state.
- Notifies that transporter arrived and entered keys.
- Stores the verification hash entered by transporter.
- Change state to Arrived_to_destination.
- Start time window for next transporter to enter the keys.

```
function KeysEnteredByTransporter(string memory keyT, string memory keyR) public OnlyTransporter {
    require(state == contractState.PackageReceivedByTansporter);
    emit ArrivedToDestination("Transporter Arrived To Destination and entered keys " , msg.sender);
    verificationHash[transporter] = keccak256(
            abi.encodePacked(keyT, keyR)
        );
    state = contractState.ArrivedToDestination;
    startEntryTransporterKeysBlocktime = block.timestamp;
}
```

VerifyKeyNextTransporter (Courier Contract):
This function ensures key is entered by next transporter and invokes verification function to match keys entered by former transporter.

- Check the contract state.
- Enter the keys and invokes the internal verification function.
- Change contract state to keys_entered.

```
function verifyKeyNextTransporter(string memory keyT, string memory keyR) public OnlyNextTransporter {
    require(state == contractState.ArrivedToDestination);
    emit BuyerEnteredVerificationKeys("Reciever entered keys, waiting for payment settlement", msg.sender);
    verificationHash[transporter2] = keccak256(
            abi.encodePacked(keyT, keyR)
        );
    state = contractState.KeysEntered;
    verification();
}
```

Internal Verification Function:

This verification function ensures that weather the keys entered by both buyer and seller are correct or there is dispute.

- Contract state verified as keys_entered.
- Values of verification hash of former transporter and next transporter must be equal.
- Settle prices for each agent.
- In case values of verification hash is not equal balance of contract is transferred to arbitrator.
- After Successful verification contract state changed to Passed_to_Next_Transporter otherwise contract state changed to aborted.

```
function verification() internal {
    require(state == contractState.KeysEntered);
    if(verificationHash[transporter] == verificationHash[transporter2]){
        emit SuccessfulVerification("Transporter and Next Transporter Keys Verified!");
        state = contractState.PassedToNextTransporter;
    }
    else {
        //trusted entity the Arbitrator resolves the issue
        emit VerificationFailure("Verification failed , keys do not match. Please solve the dispute off chain. No refunds.");
        state = contractState.DisputeVerificationFailure;
        arbitrator.transfer(address(this).balance);//all ether with the contract
        state = contractState.EtherWithArbitrator;
        emit EtherTransferredToArbitrator("Due to dispute all Ether deposits have been transferred to arbitrator ", arbitrator);
        state = contractState.Aborted;
    }
}
```

Courier Contract passes the flow to BT Contract where in Transporter passes package to the buyer(destination).

ConfirmedPackageReceived (BT Contract):
This function confirms package received to the destination.

- Check contract state.
- Saves the total delivery time.
- Notifies that package reached the destination and changed the state to package_recieved_by_buyer.

```
function ConfirmPackageReceived() public OnlyBuyer {
    require(state == contractState.waitingItemReceived);
    startdeliveryBlocktime = block.timestamp;//save the delivery time
    emit PackageKeyGivenToNextTransporter("The package reached destination", msg.sender);
    state = contractState.PackageReceivedByBuyer;
}
```

KeysEnteredByTransporter (BT Contract):
This function ensures keys are entered by transporter.

- Contract state verified as package_recieved_by _buyer.
- Transporter entered keys and internal verification function invoked to check key entered by transporter is correct or not.
- Contract state changed to Arrived_to_destination
- Start the time window for buyer to enter key

verifyKeysByBuyer (BT Contract):
This function ensures buyer entered the keys and internal verification invoked to ensure both transporter and buyer entered same keys.

- Contract state verified as Arrived_to_destination
- Buyer entered keys and waiting for settlement.
- Contract state changed to keys_entered.
- Internal Verification function invoked.

```
function KeysEnteredByTransporter(string memory keyT, string memory keyR) public OnlyTransporter {
    require(state == contractState.PackageReceivedByBuyer);
    emit ArrivedToDestination("Transporter Arrived To Destination and entered keys " , msg.sender);
    verificationHash[transporter] = keccak256(
            abi.encodePacked(keyT, keyR)
        );
    state = contractState.ArrivedToDestination;
    startEntryTransporterKeysBlocktime = block.timestamp;
}

function verifyKeysByBuyer(string memory keyT, string memory keyR) public OnlyBuyer{
    require(state == contractState.ArrivedToDestination);
    emit BuyerEnteredVerificationKeys("Reciever entered keys, waiting for payment settlement", msg.sender);
    verificationHash[buyer] = keccak256(
            abi.encodePacked(keyT, keyR)
        );
    state = contractState.KeysEntered;
    verification();
}
```

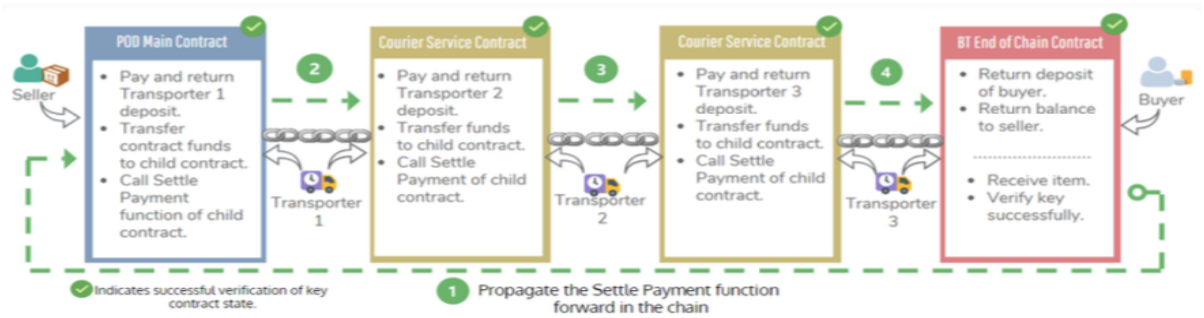
Internal Verification Function (BT_Contract):

This verification function ensures that weather the keys entered by both buyer and seller are correct or there is dispute.

- Contract state verified as buyer_key_entered.
- Values of verification hash of buyer and transporter must be equal.
- In case values of verification hash is not equal balance of contract is transferred to arbitrator.

```
function verification() internal {
    require(state == contractState.KeysEntered);
    if(verificationHash[transporter] == verificationHash[buyer]){
        emit SuccessfulVerification("Payment will shortly be settled , successful verification!");
        state = contractState.ReachedDestination;
    }
    else {
        //trusted entity the Arbitrator resolves the issue
        emit VerificationFailure("Verification failed , keys do not match. Please solve the dispute off chain. No refunds.");
        state = contractState.DisputeVerificationFailure;
        arbitrator.transfer(address(this).balance);//all ether with the contract
        state = contractState.EtherWithArbitrator;
        emit EtherTransferredToArbitrator("Due to dispute all Ether deposits have been transferred to arbitrator ", arbitrator);
        state = contractState.Aborted;
    }
}
```

Settle_Payment Function:



For payment settlement when both keys submitted by buyer and last transporter verified by internal verification function within the smart contract. Call to Main POD Contract takes place where in chain of settle payments functions of child contracts takes place.

Main POD Settle Payment Function:

This function transfers the collateral of transporter and 10% transportation fees to transporter.

- Settle funds for transporter.
- Create notification for invoking the Courier contract's settle payment function.
- Transfer the remaining funds to courier contract.

```
function getBalance() public view returns(uint){

    return address(this).balance;


}

function settle_payement() public payable {

    transporter.transfer((2*itemPrice) + ((10*itemPrice)/100));//receiver gets 10% of item price delivered
    emit SettlePayementTransferedToCourierContract('Transporter fees are settled and balance transfered to Courier Contract',msg.sender)
    courier_contract_address.transfer(address(this).balance);

}
```

Courier Contract Settle Payment Function:

This function transfers the collateral of next transporter/s and 10% transportation fees to next transporter/s account.

- Create notification for balance received by POD contract.
- Settle funds for next transporters.
- Create notification for invoking the BT contract's settle payment function.
- Transfer the remaining funds to BT contract.

```
function getBalance() public view returns(uint){
    return address(this).balance;
}



function settle_payement() public payable {
  emit deposit_from_POD_contract('Balance transfered from POD Main Contract Recieved',msg.sender);
    transporter2.transfer((2*itemPrice) + ((10*itemPrice)/100));//receiver gets 10% of item price delivered
    emit SettlePayementTransferedToCourierContract('Transporter fees are settled and balance transfered to BT_contract',msg.sender);
    BT_contract_address.transfer(address(this).balance);

}
```

BT Contract (Buyer Transporter Contract) Settle Payment Function:

This function transfers the collateral of buyer and remaining balance to seller.

- Create notification for balance received by Courier contract.
- Settle Funds for buyer.
- Settle Funds for seller.
- Create notification for Successful Payment Settlement.
- Change contract state to PaymentSettledSuccess.

```
function getBalance() public view returns(uint){
    return address(this).balance;
}


function settle_payement() public payable{
    emit deposit_from_Courier_contract('Balance transfered from Courier Contract Recieved',msg.sender);
    buyer.transfer(itemPrice);
    seller.transfer(address(this).balance);

  emit PaymentSettledSuccess('Payment Settled Successfully ',msg.sender);
  state=contractState.PaymentSettledSuccess;
}
```

In each contract get_balance function is there to fetch the current contract balance after execution of transactions.

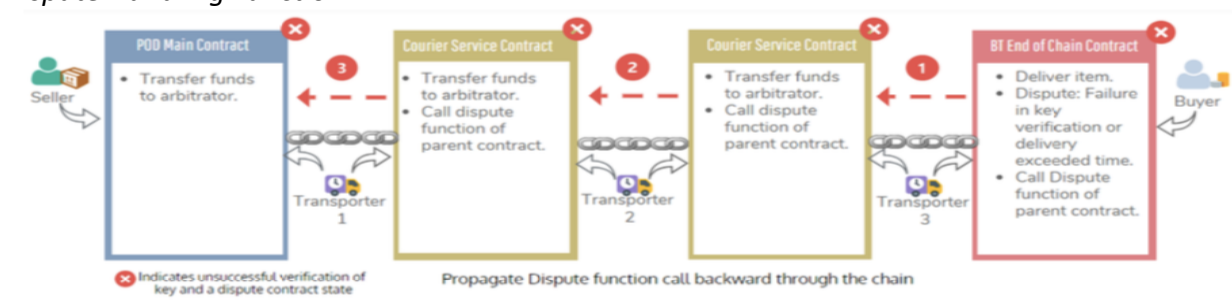
SUCCESSFUL TRANSACTION FLOW:

```
creation of POD pending...

[vm] from: 0x5b3...3bf20 to: POD.(constructor) value: 0 wei data: 0x608...00000 logs: 0 hash: 0xc99...28841 Debug

transact to POD.SignTermsAndConditions pending ...

[vm] from: 0x292...3b6db to: POD.SignTermsAndConditions() 0x61a...6dfec value: 2000000000000000000 wei data: 0x577...da7d2
logs: 2 hash: 0xe57...1618b Debug

transact to POD.SignTermsAndConditions pending ...

[vm] from: 0x607...98d3a to: POD.SignTermsAndConditions() 0x61a...6dfec value: 2000000000000000000 wei data: 0x577...da7d2
logs: 2 hash: 0xc93...e0885 Debug

transact to POD.SignTermsAndConditions pending ...

[vm] from: 0x361...8a4dd to: POD.SignTermsAndConditions() 0x61a...6dfec value: 2000000000000000000 wei data: 0x577...da7d2
logs: 2 hash: 0x4f0...e5ab1 Debug

transact to POD.createPackageAndKey pending ...

[vm] from: 0x292...3b6db to: POD.createPackageAndKey() 0x61a...6dfec value: 0 wei data: 0x3f8...9301e logs: 1
hash: 0xcfc...a80aa Debug

creation of Courier pending...

[vm] from: 0x5b3...3bf20 to: Courier.(constructor) value: 0 wei data: 0x608...00000 logs: 0 hash: 0xfbd...aa611 Debug

transact to Courier.SignTermsAndConditions pending ...

[vm] from: 0xecb...8ea6d to: Courier.SignTermsAndConditions() 0xefc...1ee7f value: 2000000000000000000 wei
data: 0x577...da7d2 logs: 2 hash: 0x7ca...2615f Debug

transact to Courier.ConfirmPackageReceived pending ...

[vm] from: 0xecb...8ea6d to: Courier.ConfirmPackageReceived() 0xefc...1ee7f value: 0 wei data: 0xc65...c0445 logs: 1
hash: 0x6ff...a9eb7 Debug

transact to Courier.KeysEnteredByTransporter pending ...

[vm] from: 0x607...98d3a to: Courier.KeysEnteredByTransporter(string,string) 0xefc...1ee7f value: 0 wei
data: 0xea3...00000 logs: 1 hash: 0x9b8...fad58 Debug

transact to Courier.verifyKeyNextTransporter pending ...

[vm] from: 0xecb...8ea6d to: Courier.verifyKeyNextTransporter(string,string) 0xefc...1ee7f value: 0 wei
data: 0x480...00000 logs: 2 hash: 0xff4...fadf6 Debug
```

```
[vm] from: 0xecb...8ea6d to: Courier.verifyKeyNextTransporter(string,string) 0xefc...1ee7f value: 0 wei
data: 0x480...00000 logs: 2 hash: 0xff4...fadf6                                                    Debug

creation of BT_contract pending...

[vm] from: 0x5b3...3bf20 to: BT_contract.(constructor) value: 0 wei data: 0x608...00000 logs: 0 hash: 0x0a4...9f3ed   Debug

transact to BT_contract.ConfirmPackageReceived pending ...

[vm] from: 0x361...8a4dd to: BT_contract.ConfirmPackageReceived() 0x53c...fccd7 value: 0 wei data: 0xc65...c0445 logs: 1
hash: 0xa74...1b078                                                                                Debug

transact to BT_contract.KeysEnteredByTransporter pending ...

[vm] from: 0xecb...8ea6d to: BT_contract.KeysEnteredByTransporter(string,string) 0x53c...fccd7 value: 0 wei
data: 0xea3...00000 logs: 1 hash: 0x4b9...f59d5                                                    Debug

transact to BT_contract.KeysEnteredByTransporter pending ...

[vm] from: 0xecb...8ea6d to: BT_contract.KeysEnteredByTransporter(string,string) 0x53c...fccd7 value: 0 wei
data: 0xea3...00000 logs: 1 hash: 0x4b9...f59d5                                                    Debug

transact to BT_contract.verifyKeysByBuyer pending ...

[vm] from: 0x361...8a4dd to: BT_contract.verifyKeysByBuyer(string,string) 0x53c...fccd7 value: 0 wei data: 0xafc...00000
logs: 2 hash: 0x42a...3dfab                                                                        Debug

transact to POD.settle_payement pending ...

[vm] from: 0x361...8a4dd to: POD.settle_payement() 0x61a...6dfec value: 0 wei data: 0x58f...44f8b logs: 1
hash: 0x207...f637c                                                                                Debug

transact to Courier.settle_payement pending ...

[vm] from: 0x583...40225 to: Courier.settle_payement() 0xefc...1ee7f value: 390000000000000000 wei data: 0x58f...44f8b
logs: 2 hash: 0x161...a810c                                                                        Debug

transact to BT_contract.settle_payement pending ...

[vm] from: 0xdd8...92148 to: BT_contract.settle_payement() 0x53c...fccd7 value: 380000000000000000 wei
data: 0x58f...44f8b logs: 2 hash: 0x14d...a6705                                                    Debug
```

```
(16) data: 0x58f44f8b

[vm] from: 0xdd8...92148 to: BT_contract.settle_payement() 0x085...a8194 value: 3800000000000000000 wei
data: 0x58f...44f8b logs: 2 hash: 0xcfe...09c89

status                  0x1 Transaction mined and execution succeed
transaction hash        0xcfe1e9aabc9e515a428baaaf7cd99ac8fb1b260855c64e91acfa1029d7d09c89
from                    0xdd870fa1b7c4700f2bd7f44238821c26f7392148
to                      BT_contract.settle_payement() 0x085e6eae1bbcba48d48d4b1bb522e1bac4aa8194
gas                     5000000 gas
transaction cost        48889 gas
execution cost          27617 gas
hash                    0xcfe1e9aabc9e515a428baaaf7cd99ac8fb1b260855c64e91acfa1029d7d09c89
input                   0x58f...44f8b
decoded input           {}
decoded output          {}
logs                    [ { "from": "0x085e6eae1bbcba48d48d4b1bb522e1bac4aa8194", "topic":
                        "0x76cf95d36c20cbb71d764eb809abf969efe61a3be4402a836702438333a46879", "event":
                        "deposit_from_Courier_contract", "args": { "0": "Balance transfered from Courier Contract
                        Recieved", "1": "0xdD870fA1b7C4700F2BD7f44238821C26f7392148", "info": "Balance transfered from
                        Courier Contract Recieved", "entityAddress": "0xdD870fA1b7C4700F2BD7f44238821C26f7392148",
                        "length": 2 } }, { "from": "0x085e6eae1bbcba48d48d4b1bb522e1bac4aa8194", "topic":
                        "0x03cdc74e9e0cec0adeb917f9e6d213c8cc8f3f8d4a2416e9abc0ebf5d5eb0eb1", "event":
                        "PaymentSettledSuccess", "args": { "0": "Payment Settled Successfully ", "1":
                        "0xdD870fA1b7C4700F2BD7f44238821C26f7392148", "info": "Payment Settled Successfully ",
                        "entityAddress": "0xdD870fA1b7C4700F2BD7f44238821C26f7392148", "length": 2 } } ]
value                   3800000000000000000 wei

>
```

Dispute Handling Function:



For Dispute Handling when last transporter reaches out to buyer for item delivery keys don't matched and it invokes chain of parent contract's dispute function to transfer funds stored in that contract to arbitrator till it reaches to Main_POD contract.

BT_Contract Dispute Function:

The function ensures after delivery item verification of keys by buyer and transporter must be done. It invokes the function call to parent contract.

- Check state of contract as Dispute_Verification_Failure
- Create notification of parent contract's dispute function invoked.
- Change states to aborted.

```
function dispute() public{
      require(state==contractState.DisputeVerificationFailure);

        emit Dispute_Function_call_of_parent('Dispute Function of parent contract invoked',arbitrator);
        state = contractState.Aborted;


}
```

Courier Dispute Function:

The function invoked by child contract BT; transfers the contract balance to arbitrator.

- Transfer funds to arbitrator.
- Change state of contract to EtherWithArbitrator.
- Create notification of ether with arbitrator.
- Create notification of parent contract's dispute function invoked and change state to aborted.

```
    function dispute() public{

        arbitrator.transfer(address(this).balance);//all ether with the contract
        state = contractState.EtherWithArbitrator;
        emit EtherTransferredToArbitrator("Due to dispute all Ether deposits have been transferred to arbitrator ", arbitrator);
        emit Dispute_Function_call_of_parent('Dispute Function of parent contract invoked',arbitrator);
        state=contractState.Aborted;

    }
```

POD_Main Dispute Function:

The function invoked by child contract Courier; transfers the contract balance to arbitrator.

- Transfer funds to arbitrator.
- Change state of contract to EtherWithArbitrator.
- Create notification of ether with arbitrator and change state to aborted.

```
function dispute() public {
arbitrator.transfer(address(this).balance);//all ether with the contract
state = contractState.EtherWithArbitrator;
emit EtherTransferredToArbitrator("Due to dispute all Ether deposits have been transferred to arbitrator ", arbitrator);
state = contractState.Aborted;

}
```

DISPUTE TRANSACTION FLOW:

```
[vm] from: 0x4a9...a77c6 to: BT_contract.verifyKeysByBuyer(string,string) 0x8c9...b48ee value: 0 wei data: 0xafc...00000 logs: 2 hash: 0xfa1...e7da6
Debug

status                  0x1 Transaction mined and execution succeed
transaction hash        0xfa1eeeb91a829ba5d35d747c8c7e8f05870c27e1ad67c4cd13fecf0c47fe7da6
from                    0x4a97f93cad2456c4dcd2408654c3566e53ba77c6
to                      BT_contract.verifyKeysByBuyer(string,string) 0x8c95a3687bb592c940a09d7bb2892e55c52b48ee
gas                     3000000 gas
transaction cost        61202 gas
execution cost          38778 gas
hash                    0xfa1eeeb91a829ba5d35d747c8c7e8f05870c27e1ad67c4cd13fecf0c47fe7da6
input                   0xafc...00000
decoded input           { "string keyT": "3", "string keyR": "4" }
decoded output          {}
logs                    [ { "from": "0x8c95a3687bb592c940a09d7bb2892e55c52b48ee", "topic":
                        "0xd58c2147902b2eb7ba2e7297446aad01cc528c23b8b37c0bf79b1a8dda675f80", "event":
                        "BuyerEnteredVerificationKeys", "args": { "0": "Reciever entered keys, waiting for payment
                        settlement", "1": "0x4a97F93Cad2456c4Dcd2408654C3566e53BA77c6", "info": "Reciever entered keys,
                        waiting for payment settlement", "entityAddress": "0x4a97F93Cad2456c4Dcd2408654C3566e53BA77c6",
                        "length": 2 } }, { "from": "0x8c95a3687bb592c940a09d7bb2892e55c52b48ee", "topic":
                        "0x4ed0570313882c4b1660d5a3a0fdca6fbeddba1bfca75a125dcd7e403f192442", "event":
                        "VerificationFailure", "args": { "0": "Verification failed , keys do not match. Please solve the
                        dispute off chain. No refunds.", "info": "Verification failed , keys do not match. Please solve
                        the dispute off chain. No refunds.", "length": 1 } } ]
value                   0 wei
```

```
transact to BT_contract.dispute pending ...

[vm] from: 0x4a9...a77c6 to: BT_contract.dispute() 0x8c9...b48ee value: 0 wei data: 0xf24...0f7c3 logs: 1 hash: 0x003...25691
Debug

transact to Courier.dispute pending ...

[vm] from: 0x4a9...a77c6 to: Courier.dispute() 0xe95...9d78f value: 0 wei data: 0xf24...0f7c3 logs: 2 hash: 0xef9...496e2
Debug

transact to POD.dispute pending ...

[vm] from: 0x4a9...a77c6 to: POD.dispute() 0xe97...7231f value: 0 wei data: 0xf24...0f7c3 logs: 1 hash: 0xc5e...efd24
Debug
```

[vm] **from:** 0x4a9...a77c6 **to:** BT_contract.dispute() 0x8c9...b48ee **value:** 0 wei **data:** 0xf24...0f7c3 **logs:** 1 **hash:** 0x003...25691

Debug

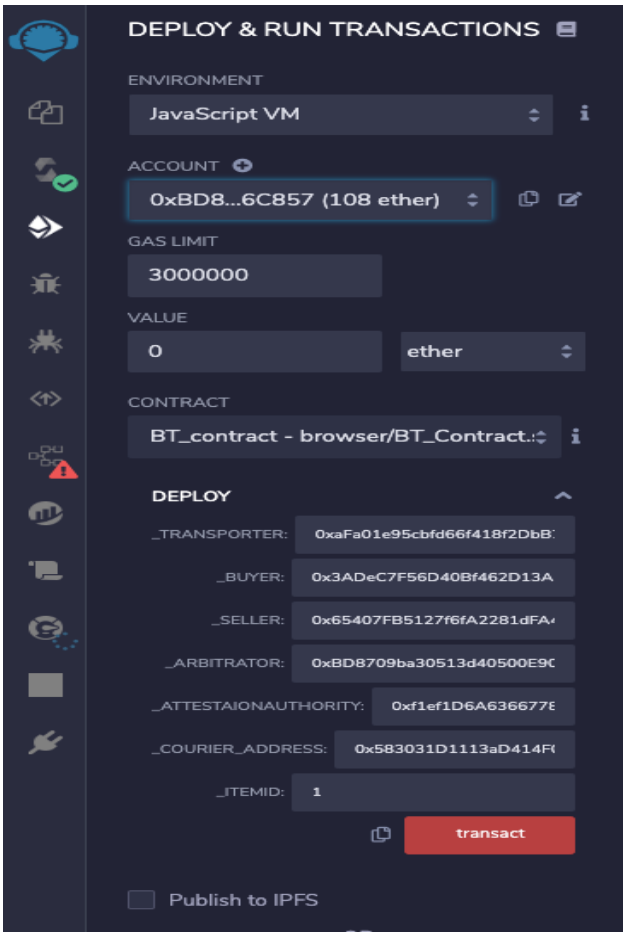
status	0x1 Transaction mined and execution succeed
transaction hash	0x003cf8974676975518dd9a1b602a894ad876a1cbd3311362fb2a6ef65a225691
from	0x4a97f93cad2456c4dcd2408654c3566e53ba77c6
to	BT_contract.dispute() 0x8c95a3687bb592c940a09d7bb2892e55c52b48ee
gas	3000000 gas
transaction cost	31321 gas
execution cost	10049 gas
hash	0x003cf8974676975518dd9a1b602a894ad876a1cbd3311362fb2a6ef65a225691
input	0xf24...0f7c3
decoded input	{}
decoded output	{}
logs	[{ "from": "0x8c95a3687bb592c940a09d7bb2892e55c52b48ee", "topic": "0xc4b6bcd7cd85fc178dfc4f737c3e752ce4cfdc5d634032ce328ca622248753a9", "event": "Dispute_Function_call_of_parent", "args": { "0": "Dispute Function of parent contract invoked", "1": "0x951Bc6787c38C74d1821f61030a894ad45762e61", "info": "Dispute Function of parent contract invoked", "entityAddress": "0x951Bc6787c38C74d1821f61030a894ad45762e61", "length": 2 } }]
value	0 wei

[vm] **from:** 0x4a9...a77c6 **to:** Courier.dispute() 0xe95...9d78f **value:** 0 wei **data:** 0xf24...0f7c3 **logs:** 2 **hash:** 0xef9...496e2

Debug

status	0x1 Transaction mined and execution succeed
transaction hash	0xef9a2bba8e52672e7f292649bfeda398f95ae131679e75560b14de77deb496e2
from	0x4a97f93cad2456c4dcd2408654c3566e53ba77c6
to	Courier.dispute() 0xe958d39c97216b45b46dc45c846931f12e99d78f
gas	3000000 gas
transaction cost	42015 gas
execution cost	20743 gas
hash	0xef9a2bba8e52672e7f292649bfeda398f95ae131679e75560b14de77deb496e2
input	0xf24...0f7c3
decoded input	{}
decoded output	{}
logs	[{ "from": "0xe958d39c97216b45b46dc45c846931f12e99d78f", "topic": "0xfc0cf60f1f5c19e05c32dfb2dff6c9823cdc87846e26a2e2335a01cb184ab2f3", "event": "EtherTransferredToArbitrator", "args": { "0": "Due to dispute all Ether deposits have been transferred to arbitrator ", "1": "0x951Bc6787c38C74d1821f61030a894ad45762e61", "info": "Due to dispute all Ether deposits have been transferred to arbitrator ", "entityAddress": "0x951Bc6787c38C74d1821f61030a894ad45762e61", "length": 2 } }, { "from": "0xe958d39c97216b45b46dc45c846931f12e99d78f", "topic": "0xc4b6bcd7cd85fc178dfc4f737c3e752ce4cfdc5d634032ce328ca622248753a9", "event": "Dispute_Function_call_of_parent", "args": { "0": "Dispute Function of parent contract invoked", "1": "0x951Bc6787c38C74d1821f61030a894ad45762e61", "info": "Dispute Function of parent contract invoked", "entityAddress": "0x951Bc6787c38C74d1821f61030a894ad45762e61", "length": 2 } }]
value	0 wei

[vm] **from:** 0x4a9...a77c6 **to:** POD.dispute() 0xe97...7231f **value:** 0 wei **data:** 0xf24...0f7c3 **logs:** 1 **hash:** 0xc5e...efd24 **Debug**

status	0x1 Transaction mined and execution succeed
transaction hash	0xc5e1cd4c33f3cb6b054a8e9efac7964218b6c9aaabf55da2ac6d1e8b4e2efd24
from	0x4a97f93cad2456c4dcd2408654c3566e53ba77c6
to	POD.dispute() 0xe97a100dd45d018d2f0747670847bfcbabf7231f
gas	3000000 gas
transaction cost	40679 gas
execution cost	19407 gas
hash	0xc5e1cd4c33f3cb6b054a8e9efac7964218b6c9aaabf55da2ac6d1e8b4e2efd24
input	0xf24...0f7c3
decoded input	{}
decoded output	{}
logs	[{ "from": "0xe97a100dd45d018d2f0747670847bfcbabf7231f", "topic": "0xfc0cf60f1f5c19e05c32dfb2dff6c9823cdc87846e26a2e2335a01cb184ab2f3", "event": "EtherTransferredToArbitrator", "args": { "0": "Due to dispute all Ether deposits have been transferred to arbitrator ", "1": "0x951Bc6787c38C74d1821f61030a894ad45762e61", "info": "Due to dispute all Ether deposits have been transferred to arbitrator ", "entityAddress": "0x951Bc6787c38C74d1821f61030a894ad45762e61", "length": 2 } }]
value	0 wei

Balance of Arbitrator after dispute called:
Item Price is 1 ether so total collateral deposited 6 ethers(buyer, seller and transporter) then collateral from 2nd transporter 2 ethers(next transporter), so in total 8 ethers which get deposited to arbitrator account in case of dispute, balance of account changed from 100 to 108.