Report 04/12/2017

-
- Platforms:
    - Hyperledger
        - Permissioned,
        - fast enough,
        - Provides Enterprise solution
        - Industry backing the project
        - → suitable, **first choice for prototype**
        - **Fabric and Burrow (Monax)**
        - Others are not evolved enough
    - Ethereum
        - Permissionless
        - Dependent on Gas and Ethereum price
        - → not suitable
    - IOTA
        - Not ready for use
        - No Smart Contracts
        - → not suitable yet
        - → **later might be a good alternative**
    - Bitcoin
        - Permissionless
        - → not suitable
    - ZCash
        - Better privacy
        - Very slow
        - No smart Contracts
        - → not suitable
    - UBQ
        - Ethereum with tweaks
        - Future uncertain
        - → not suitable

- Design plans:

Below is the basic architecture of the Container Trading Platform. It should represent the minimum viable product.
We define 4 different pages/actions for now (will be extended in the future). These pages will be done on a normal website, using html/css/javascript and running on AWS.

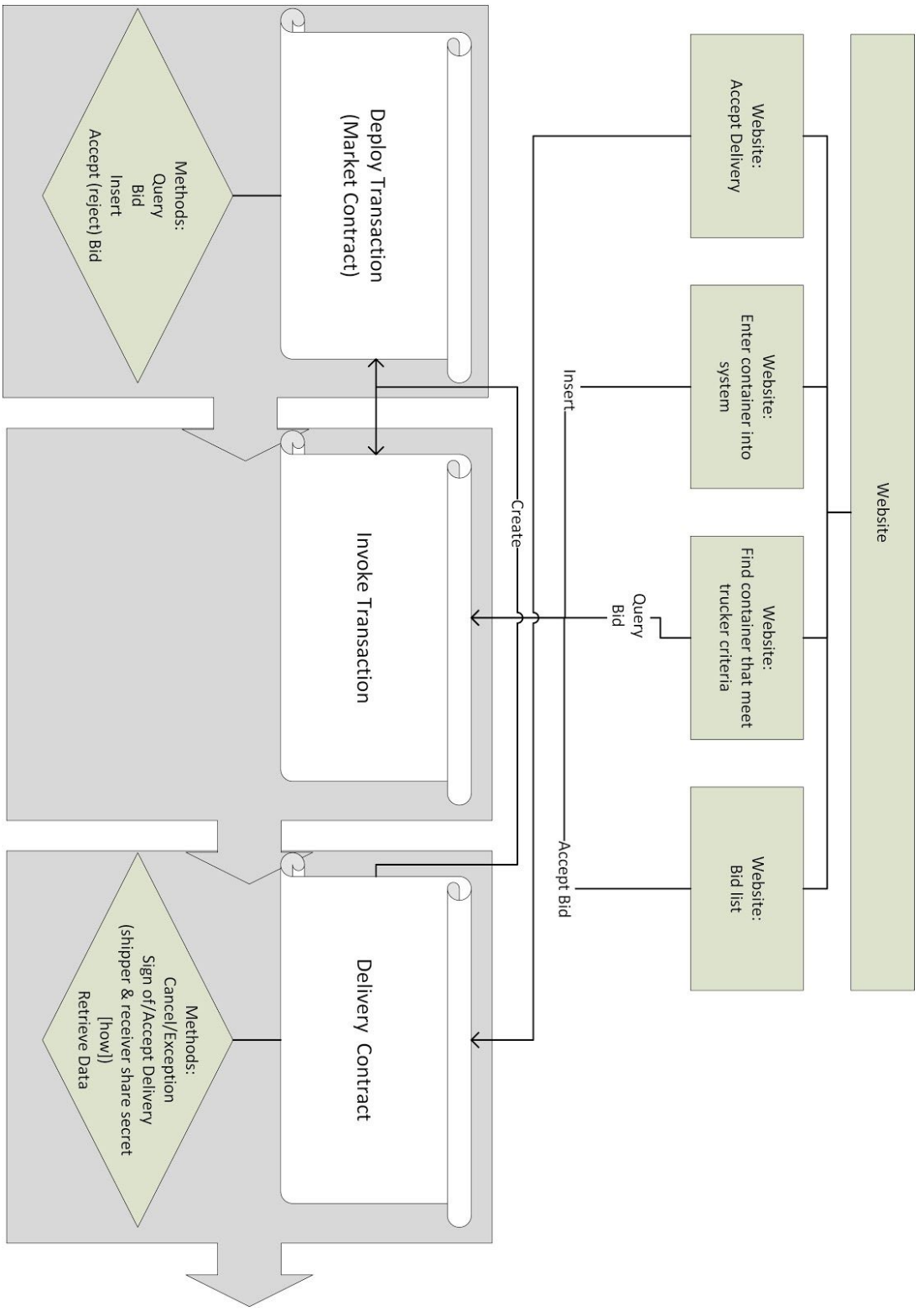
- **Insert Container**: this page will be used by the container owners to add a newly arriving container into the system. It can either be a GUI, or controlled by an external system through an API call.
- **Container List**: this is a page that truckers will use to find available containers and bid on them. Truckers will fill out a form that will contain details of their availability: truck size, time, rating, etc… Some data will be filled out automatically. Some will have to be entered manually every time. After the search is complete the trucker will be presented with a list of containers and their public information. Then they will be able to bid on these containers on the same page. Private container information, like id, exact destination, shipping company, etc. are secret at this point.
- **Accept delivery**: once the delivery of a container is finished, this page is used by the receiver (which could be a third party), to confirm the delivery. If it is confirmed the trucker automatically receives a positive binary rating. If it was not delivered on time he receives a negative rating.
- **Bid List:** this page is for the Container owner, to compare the bids various truckers made on his container and finally select one. Automatically all others bidders receive a reject message. The accepted Trucker gets immediately hidden from other bid lists he was on. Optionally we can include the following scenario: If he places bids on different time slots and wants to serve both of them (assuming he gets the first contract) because the first one requires little time he mustn't be excluded from the other bidding.

Hyperledger has two types of transactions: deploy transactions and invoke transactions. Deploy transactions are basic smart contracts that can contain data/state/program code. Invoke transactions are used to call other previously deployed deploy transactions. We will be using them both as described below.

- **Marketplace Contract:** this will be a deploy transaction that will contain public interface methods and the entire dataset of currently available containers and bids.
    - **Insert:** this method is called from the Insert Container page by the container owner. It will add a new container into the state of the marketplace contract.
    - **Query:** this method will be called from the container list page from the website. It will take several parameters as arguments and return a list of containers that match the criteria.
    - **Bid:** this method is also called from the container list page. It will accept encrypted information of the trucker, bid amount and container id(internal, private). Afterwards it will update the state of the container with the bid information and maybe inform the container owner based on an algorithm.
    - **Accept/reject bid:** this is called form the bid list page. The container owner can accept a bid for the container. When this method is called, it will update contract state that this container is no longer available. It will send an event to

all other bidders of this container and create a new Delivery Contract. This new contract will only be accessible by the two parties involved: the trucker and the shipper. At this point full information is revealed and both parties are notified.

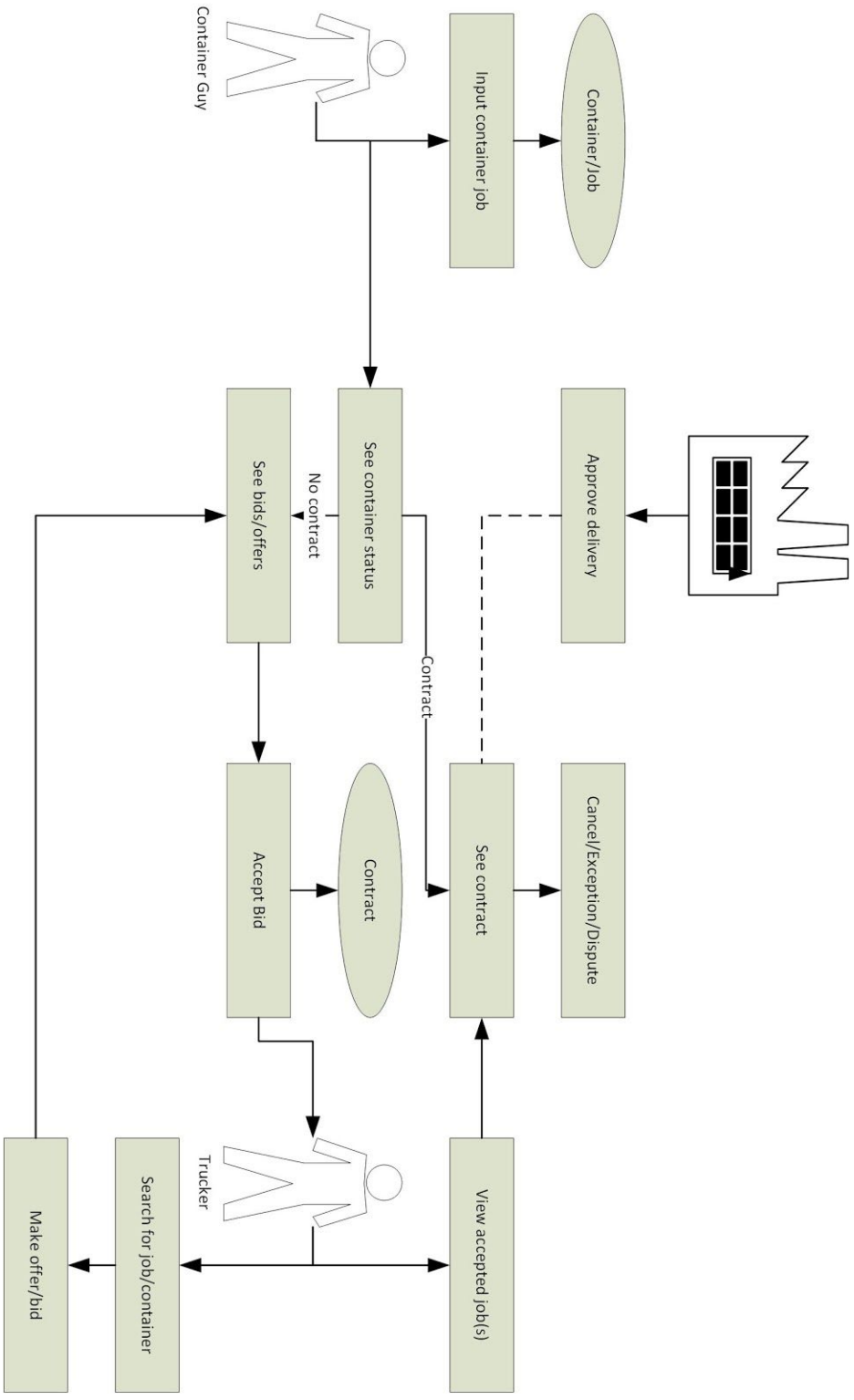
- **Delivery Contract:** this contract will contain information of a single delivery. It will have info of the exact destination of the container, its slot, ID, trucker's name and any other info that is needed. It will have several methods that can be call through another Invoke Transaction from the website.
    - **Cancel/Raise Exception:** If for any reason this delivery cannot be completed by either of the two parties, this method is called with certain parameters. Once it is called, the contract will notify both parties and log this exception on the blockchain. In certain scenarios it could also call the Marketplace Contract.
    - **Sign off:** this method is called when the delivery has been completed. It can either be triggered by the owner of the container or a third party.
    - **Retrieve Data**: this method will return the entire history and information of the contract. It can be used for printing invoices, handling disputes and other reasons.



- System use cases:

The diagram shows the use cases of all actors in the system:

- “Container Guy” - entity which wants a container delivered ,
- “Trucker” - entity which performs the delivery of the container,
- “Third party” - entity which accepts the delivery of the container by “Trucker”.



- Known issues that we are aware of, but didn't solve yet:
    - Hiding truckers identities,
    - Money - payments using blockchain,
    - Possible security issues.
    - Details on rating system