Towards Understandable Smart Contracts

Mark Moir, Architect, Oracle Labs

Joint work with: Harold Carr, Davin Fifield, Chris Flemming, Bill Xie

Symposium on Distributed Ledger Technology
Griffiths University, Gold Coast, Australia
June 2017

Safe Harbor Statement

The preceding is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

Distributed Ledgers and Smart Contracts

- Distributed Ledger Technology (aka “blockchain”) enables trustworthy sharing and updating of tamper-proof data between untrusting parties
  - Currencies, digital (or tokenized) assets, …
- Smart contracts enable flexible and expressive logic to govern updates
- Combination enables exciting new use cases
  - Payment triggered by agreed conditions (on-time delivery, IoT device confirms shipping conditions within agreed parameters)
  - …
  - Decentralized Autonomous Organization (DAO)
    - Crowd funding, proposals approved via voting by token holders, no (biased, corruptible) humans
    - Proposals can generate income for the DAO, effectively a decentralized investment fund

ORACLE

“The” DAO

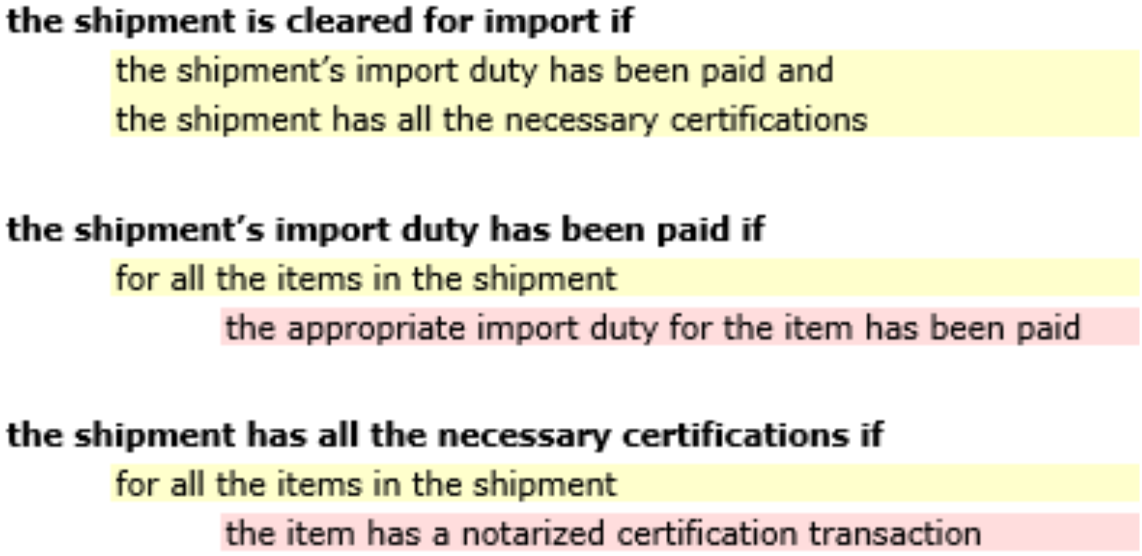
- Launched on public Ethereum platform in May 2016
- Raised equivalent of > US $160M
- Quickly attacked, resulting (kind of) in loss of > US $50M
- (Subset of) community rallied to create “hard fork” that special-cased recovery of lost funds
- Now we have two Ethereums (Etherea?)

Ooops! What went wrong with the DAO?

- Ethereum platform behaved correctly, as specified
- DAO smart contract contained multiple bugs, did not reflect authors' intent or (presumably) investors' understanding
- Written in Solidity (new Javascript-like language) for Ethereum VM
- Choice and design of programming language can be debated, but…
- … **any** general-purpose language is difficult to understand for most people
- If experts did not catch bugs, what hope is there for a human (e.g. potential investor) to understand exactly **what** s/he is trusting?

Oracle ® Policy Automation (OPA)

- Express business rules and policies in **human-readable** language
- Rules compiled to machine-readable format
- OPA engine applies rules to data
  - Input attributes + inference => Goal attributes



the shipment is cleared for import if

- the shipment’s import duty has been paid and
- the shipment has all the necessary certifications

the shipment’s import duty has been paid if

- for all the items in the shipment
  - the appropriate import duty for the item has been paid

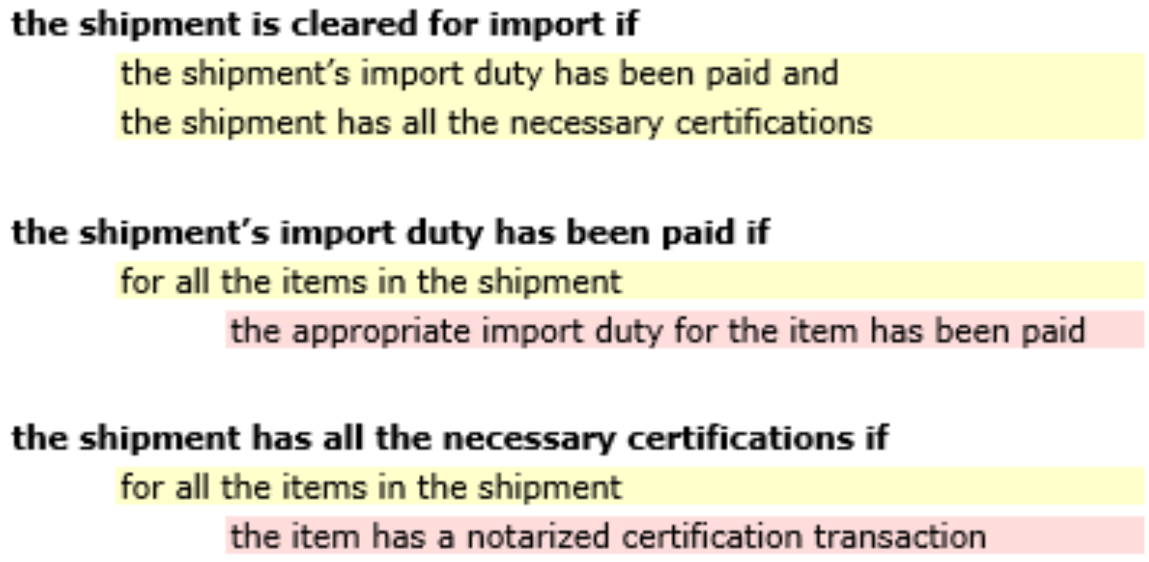
the shipment has all the necessary certifications if

- for all the items in the shipment
  - the item has a notarized certification transaction

Top-Level Conditions for Approving Imports

Oracle ® Policy Automation (OPA)

- Express business rules and policies in **human-readable** language
- Rules compiled to machine-readable format
- OPA engine applies rules to data
  - Input attributes + inference => Goal attributes
- Typical use: centralized database



the shipment is cleared for import if

- the shipment's import duty has been paid and
- the shipment has all the necessary certifications

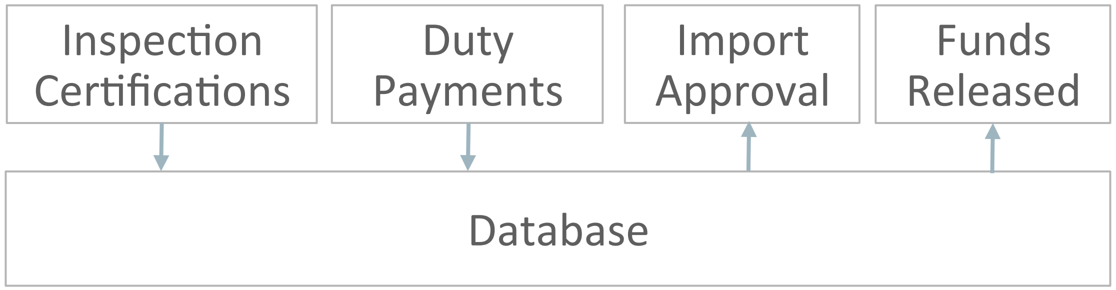
the shipment's import duty has been paid if

- for all the items in the shipment
  - the appropriate import duty for the item has been paid

the shipment has all the necessary certifications if

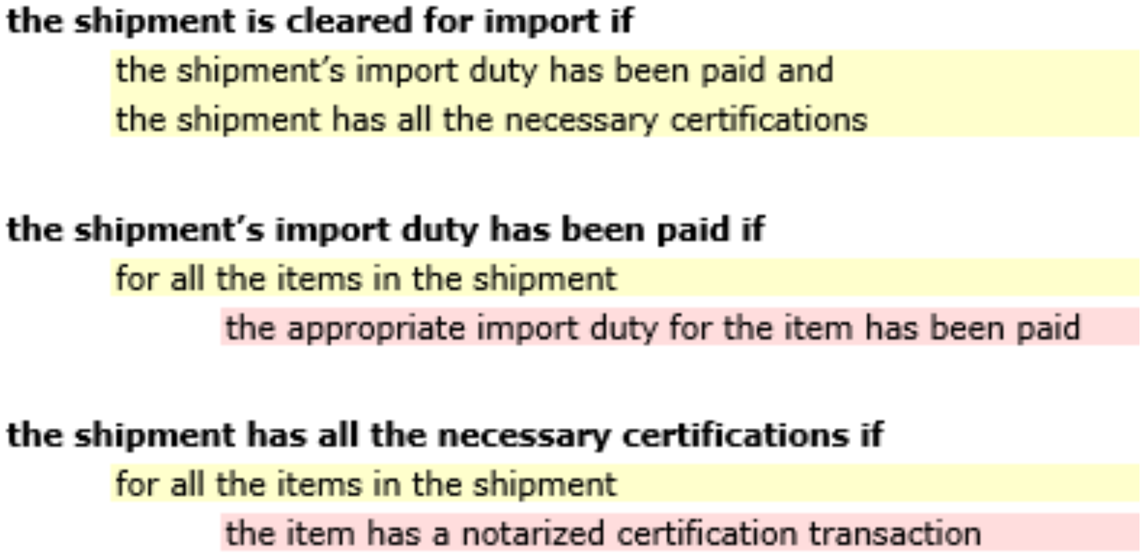
- for all the items in the shipment
  - the item has a notarized certification transaction

Top-Level Conditions for Approving Imports



Oracle ® Policy Automation (OPA)

- Express business rules and policies in **human-readable** language
- Rules compiled to machine-readable format
- OPA engine applies rules to data
  - – Input attributes + inference => Goal attributes
- Typical use: centralized database
- We explore integration to enable OPA rules as smart contracts

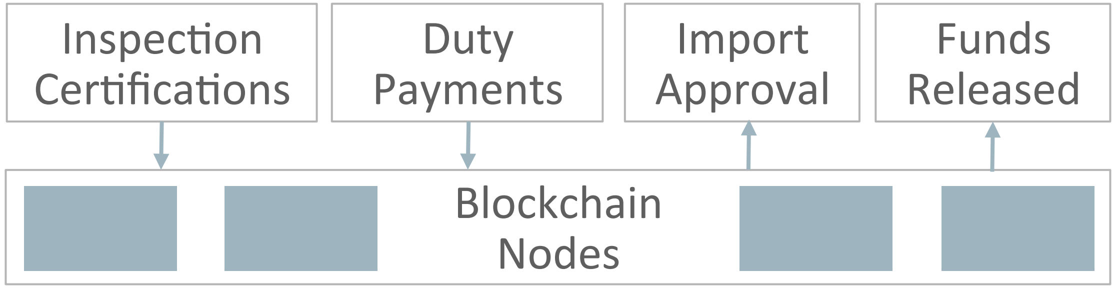


```
the shipment is cleared for import if
        the shipment's import duty has been paid and
        the shipment has all the necessary certifications

the shipment's import duty has been paid if
        for all the items in the shipment
                the appropriate import duty for the item has been paid

the shipment has all the necessary certifications if
        for all the items in the shipment
                the item has a notarized certification transaction
```

Top-Level Conditions for Approving Imports



OPA as Smart Contract Language?

- Human readable
- Deterministic
- Guaranteed to terminate
  - may not reach conclusion if configured to stop too soon, but deterministic
- These qualities address significant challenges for smart contracts

Juno

- Open source blockchain platform
- Implemented in Haskell
- “BFT hardened” Raft protocol for consensus
- Built-in DSL for money transfers

Generalizing Juno

- We modified Juno to support “pluggable” smart contract engines
  - Engine defines ledger state type, initial state, command processor for updating state
- Smart contract engine for OPA
  - State: key-value map
  - Command processor:
    - Sends input attributes and metadata from application command to “servlet”
    - To be continued…

OPA Servlet

- Has registered OPA rule set
- Receives a request containing:
  - Rule set ID
  - Input attributes
  - Scope ID (e.g., identify customer, case, etc.)
- Invokes OPA engine via Java API, provides input attributes
- Receives result from OPA engine, including inferred goal attributes(s)
- Produces transaction and sends it back to Juno:
  - read set contains input attributes, write set contains goal attributes
  - keys derived from data model, attribute names, scope ID, etc.

Generalizing Juno (continued)

- We modified Juno to support “pluggable” smart contract engines
  - Engine defines ledger state type, initial state, command processor for updating state
- Smart contract engine for OPA
  - State: key-value map
  - Command processor:
    - Sends input attributes and metadata from application command to “servlet”
    - Receives results from servlet in form of a transaction (read and write sets)
    - Validates read set against key-value store; if successful…
    - … updates key-value store based on write set

Benefits

- Tamper-proof record of transaction in blockchain
- Transactions driven by human-understandable policies
- OPA can “explain” reasoning, providing valuable audit trail
  - Can be reproduced on demand after the fact, as OPA is stateless
- No trusted party, no single point of failure

Concluding Remarks

- This is an experimental prototype for research
- Demonstrates feasibility of integrating human-readable policies with a blockchain platform
- More work needed to achieve/assess practicality for real use cases

Questions?

mark.moir@oracle.com

ORACLE®