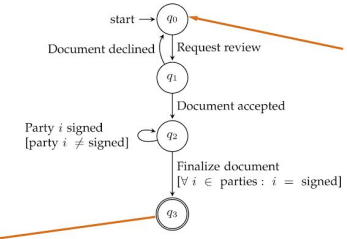
LTO Network

Hybrid Blockchain for Decentralized Workflows

Tech details

LTO Network - Quick overview

Upper layer: permissionless private chains. Every decentralized workflow utilises a different miniature private chain. Events are anchored as Proof of Existence on the bottom permissionless public blockchain layer.



Every process (basically any inter-party interaction) can be visualized as BPM and perceived as a decentralized workflow – where logic is expressed as extended deterministic Finite State Machine.

Bottom layer: permissionless public blockchain based on BitcoinNG. Uses new Leased Proof of Importance staking. Use case: functional anchoring.

Nodes: consist of user interface, application services, queuer. Essentially clusters of Docker containers. Every party has a node.

LTO Network

Blockchain for decentralized workflows

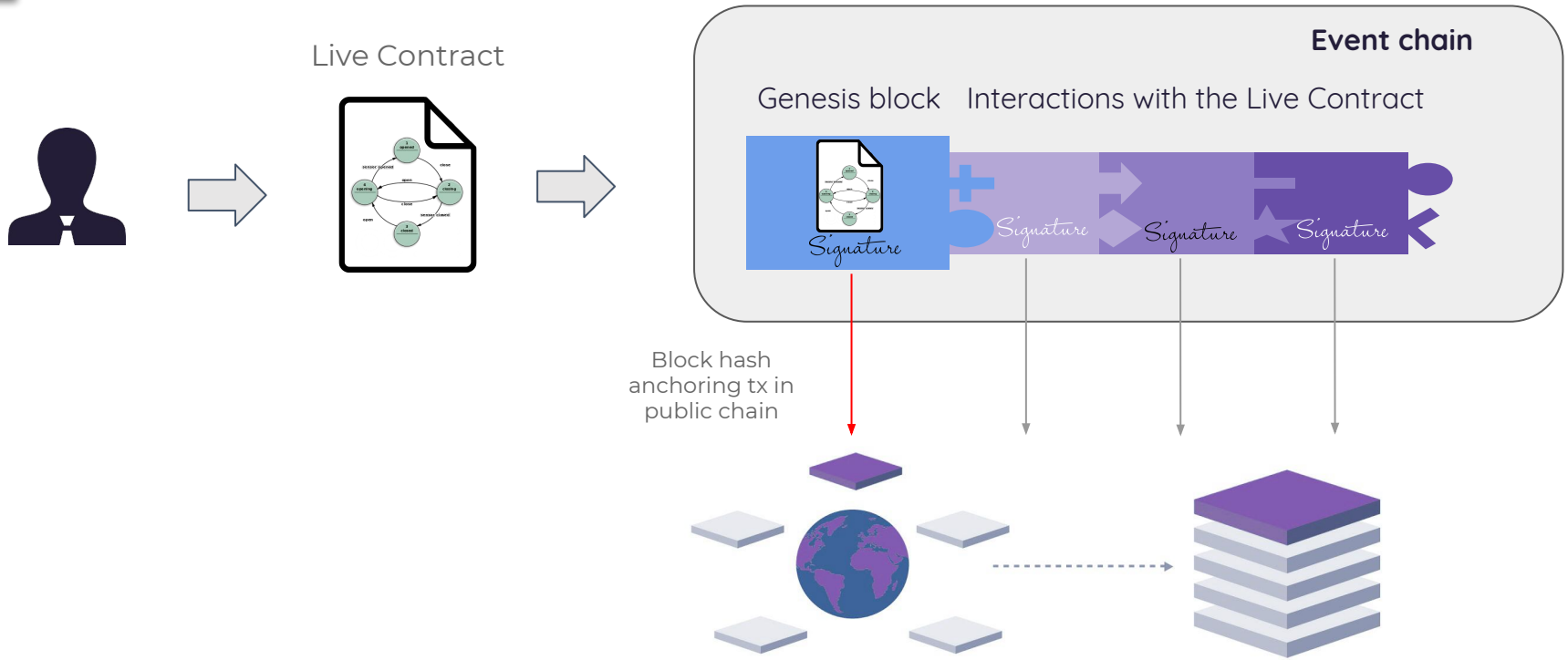
LIVE CONTRACT

Similar to Smart Contracts, Live Contracts provide a dynamic method for defining logic on the blockchain. However, the purpose of a Live Contract is not to just determine state, but to actively instruct humans and computers about steps in a workflow.

Simple integration allows it to track the state of other chains or databases via API. This can contrusct complex processes – and even integrate Smart Contracts if needed. Next to that, Live Contracts can make any other public blockchain GDPR compliant.

Integration possibilities are endless: supply chain, KYC, any legal contracts. Essentially it is interactive logic on blockchain.

Solving the problems of permissionless private chains



Setting up a node

Requirements

- Docker
- Internet connection

Setup

- Change settings in ‘docker-composer.yml’ e.g. your port (optionally)
- run docker-compose up
- Goto http://localhost:3000

LTO Node Concepts

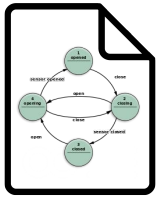
The LTO Node has 3 concepts you need to understand:

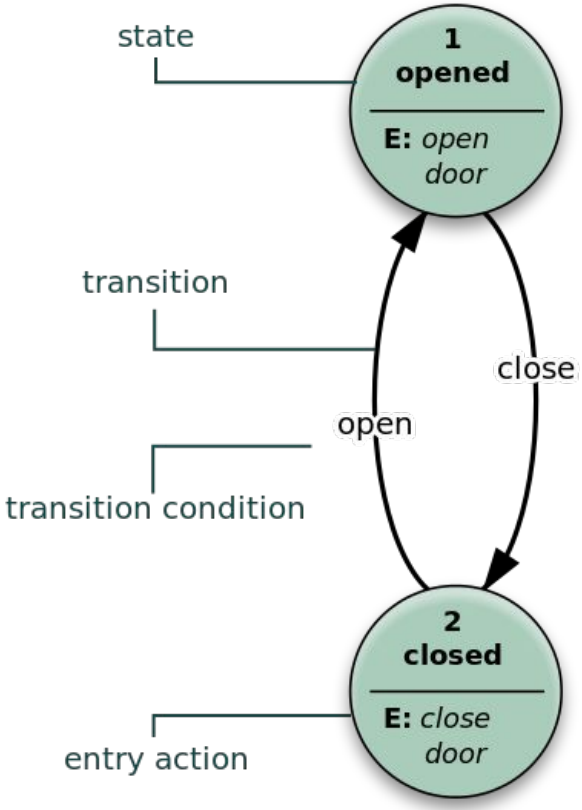
- Scenario
- Event Chain
- Process

LTO Node Concepts - Scenario

A scenario is described using a Finite State Machine

Live Contract

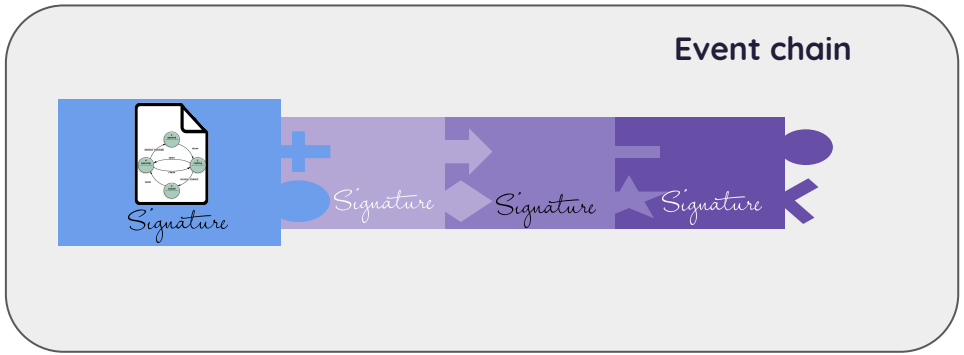




LTO Node Concepts - Event Chain

Event Chain stores 3 concepts:

1. Events, the actual event on the chain.
2. Identities, who is allowed to interact with the chain
3. Resources, which resources are linked to the chain. Resources can be things like scenarios, processes, documents, etc.



LTO Node Concepts - Process

Process is an instantiated scenario. It describes:

- current state
- possible future states
- past actions
- actors who can interact with the process
- data stored in the process in the form of assets

Building a workflow scenario (docs)

Deterministic concepts
In order for workflow to work decentralized all the concepts need to be deterministic. To make these concepts deterministic we use 2 important concepts:

1. JSON Schema: every concept within a scenario or event chain needs to be described using a json schema. (docs)
2. Deterministic Resource Identifiers: every resource used within a process needs to have a unique a reproducible id. (docs)

Building a workflow scenario or live contract (docs)

A scenario is made up out the following concepts:

- actors: who performing which action (docs)
- state: in which states can process be (docs)
- action: which actions can be performed to lead to a new state (docs)
  - normal action (docs)
  - http trigger (docs)
  - nop trigger (docs)
  - event trigger (docs)
  - custom trigger (docs)
- asset: Data to be stored in the process (docs)

Workflow concepts - Actor

Items in the actors property of the **scenario** aren't actor objects, but JSON schemas defining the properties available for the actor once instantiated in the process.

The custom schema for actors should extend the basic actor schema, which means it must at least have a title and identity and property.

```yaml
actors:
  employer:
    type: object
    title: Employer
    properties:
      title:
        type: string
      name:
        type: string
      email:
        type: string
        format: email
      identity:
        $ref: "https://specs.livecontracts.io/v0.2.0/identity/schema.json#"
  employee:
    $ref: "https://specs.livecontracts.io/v0.2.0/actor/schema.json#employee"
```

Workflow concepts - Asset

Items in the assets property of the **scenario** aren't asset objects, but JSON schemas defining the properties available for the asset once instantiated in the process.

Assets don't have any required properties, but it's common for an asset to have a title property. If this property exist, it's copied from the JSON schema.

```yaml
assets:
  contract:
    properties:
      id:
        type: string
        format: uri
      name:
        type: string
  book:
    properties:
      id:
        type: string
        format: uri
      title:
        type: string
      isbn:
        type: string
        pattern: "^\\d{13}$"
```

Workflow concepts - State

All the possible states are described in the states section. Each state describes through which action it can transition to a new state.

```
:initial:
    action: complete
    transitions:
      - response: ok
        transition: :success
      - response: cancel
        transition: :failed
```

Workflow concepts - Action

An action is something that can be performed by actor or the node of an actor. An action may trigger a state transition and / or may update the process projection.

```
issue:
  "$schema": https://specs.livecontracts.io/v0.2.0/action/schema.json
  actor: issuer
  responses:
    ok:
      title: Issued new license
      update:
      - select: assets.license
      - select: assets.available
        projection: "{shipments: shipments, quantity: quantity}"
```

There are 2 different action types:

1. Normal action: performed by receiving a response event.
2. Triggered action: performed are performed by the node itself. There are 3 types of trigger:
    a. Event trigger
    b. HTTP trigger
    c. NOP trigger

Workflow concepts - Action - Event Trigger

With an event trigger action the node itself will add an event to chain and sign it with the private key of the node itself.

Example

```
"$schema": https://specs.livecontracts.io/v0.2.0/action/event/schema.json#
actor: issuer
body:
  "$schema": https://specs.livecontracts.io/v0.2.0/identity/schema.json#
  id: !ref actors.license_holder.id
  signkeys: !ref actors.license_holder.signkeys
  node: !ref actors.license_holder.node
responses:
  ok:
    display: never
  error:
    display: always
    title: Failed to add the license holder
```

Workflow concepts - Action - HTTP Trigger

An HTTP trigger will send an http request to a configured url

Example

```yaml
$schema: https://specs.livecontracts.io/v0.2.0/action/http/schema.json#
title: Step1
url: http://example.com
method: "POST"
headers:
  Content-Type: "application/json"
data:
  foo: bar
actor: system
```

Workflow concepts - Action - NOP Trigger

No operation (NOP) trigger is able to manipulate data within the process. So it allows you to update asset or actors information.

Example

```
"$schema": https://specs.livecontracts.io/v0.2.0/action/nop/schema.json#
actor: issuer
default_response: approve
trigger_response:
  "<if>":
    condition:
      "<apply>":
        input:
          available:
            "<ref>": assets.available
        query: available.shipments > `0`
    then: approve
    else: deny
responses:
  approve: {}
  dany: {}
```

Interacting with the Live Contract

Interaction with the process is done through the event chain. So if an actor wants to respond to an action he needs to add an event to the chain and send the chain to the node.

Starting a process is done through a certain set of events.

Instantiation

- An identity event of the initiator of the process is added. This way the initiator is able to added new events to chain.
- The scenario event is added to the chain, so the node can instantiate the scenario. If the scenario is already stored through an other event this event can be skipped.

Initialization

- A process event is then added to the initialize the scenario.
- Now that the process has be initialized the first response event can be added to the chain to perform the first action.

Start

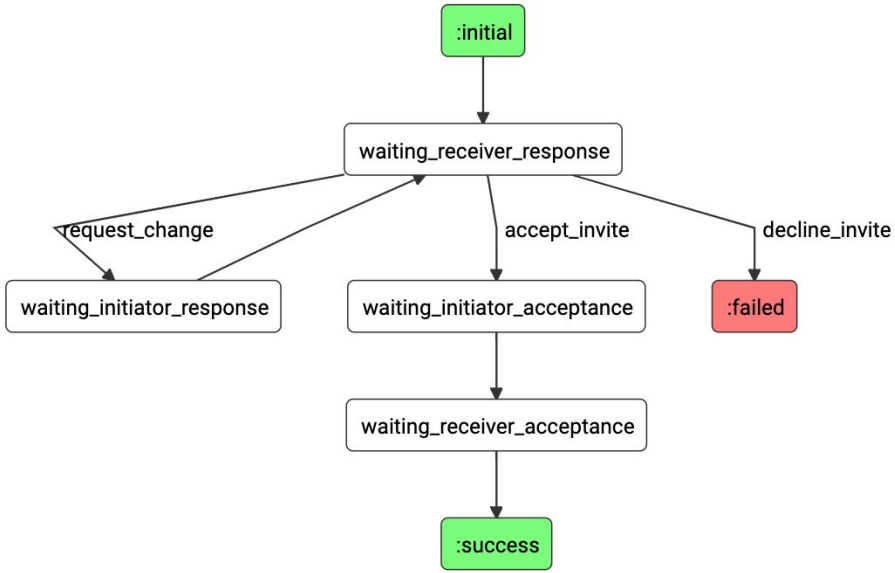
- Send in the first response to your workflow to jump to the next state.

Building a workflow

Today's assignment

- Build a simple NDA workflow, where 2 individuals agree on not disclosing a specified piece of information.

FSM:



Building a workflow - tools

In this github repository you will find: https://github.com/legalthings/lto-deepdive

- docker-compose.yml to run your node
- Bootstrap code that will help build the workflow
- this presentation for reference

Documentation you can find here: https://docs.livecontracts.io