IOTA

Version 1.0 - 23/01/2019 - Michele Nati & Jens Munch Lund-Nielsen

# *Disclaimer*

*The IOTA Foundation (IF) is a non-profit organization supporting the development and adoption of the IOTA Tangle, a permissionless Distributed Ledger Technology (DLT), particularly suitable for creating trusted information and value sharing across multi-stakeholder ecosystems. IOTA technology is open source. IF's agile approach to solutions creation leverages on identifying industry problems with real stakeholders, building PoCs and collaboratively testing, validating or refining assumptions.*

> A blueprint is a document explaining how IOTA technology is used to solve real problems and to support well-defined business needs. It should work as a reference for others to replicate and deploy the developed solution into similar systems and to test different business models.

*A blueprint is generally easy to read and understand, but nevertheless, it provides enough clarity on how easy it would be to integrate the described solution into other systems. At the same time, it contains links to more technical resources available to developers.*

# Introduction

This blueprint describes the IOTA Trade Proof of Concept (PoC). The Trade PoC is a tool that leverages IOTA technologies to address a set of common problems affecting international trade of goods.

Global trade involves a various number of assets and actors, the management of which requires access to information collected and stored in various proprietary systems. Integration of these systems is proven to be a daunting task that requires to address trust issues related to the integrity of shared data.

The document is organized as follows:

- We first describe the problem worth solving, underpinning the technical as well as the business challenges associated to it and the benefits deriving from developing a novel solution;
- We then present a technical architecture showing how this PoC leverages IOTA Tangle and other IOTA technologies and how it can be integrated with existing systems. The proposed architecture aims to provide an implementation pattern for integrating IOTA with existing legacy systems.
- We finally describe the source code showing used implementation patterns used, replicable for the integration of IOTA in similar contexts. We also provide links to accessible resources useful to simplify the deployment of the presented PoC.

## Use case

This blueprint describes how a new *data exchange layer*, based on IOTA distributed ledger technology, can be developed in order to address all the barriers existing in cross-border trading.

Cross-border trading involves a selected number of actors, including but not limited to: *shippers, forwarders, customs* and *traders*. Such actors are involved in a number of processes dealing with the following challenges: i) how trade certificates can be shared and checked for authenticity even before a shipment is initiated or when it is already on its way; ii) how the different actors handling a shipment can report its status (e.g., cleared for export, Gate-in into the port, on-board a vessel etc.); and finally iii) how the different actors can share an auditable record of the conditions of the shipped goods (temperature, location, shock, etc).

Due to the multi-stakeholder nature of these processes, simplifying them requires the creation of a data exchange layer which uses the IOTA Tangle and other IOTA technologies. IOTA DLT helps to ensure integrity of data and to maintain trust among the parties involved in the international shipment of containers goods.

Figure below shows a stakeholders map, highlighting a container journey, its different chains of custody (dotted arrows) and those stakeholders (namely custodians) eventually responsible of updating the container status and the associated shipment documents (plain arrows). In the case of international trading of goods, a container is first sent by a *shipper*. Subsequently, the container is handled by a *forwarders* until it reaches a *port operator* and later a *custom clearance* agent.

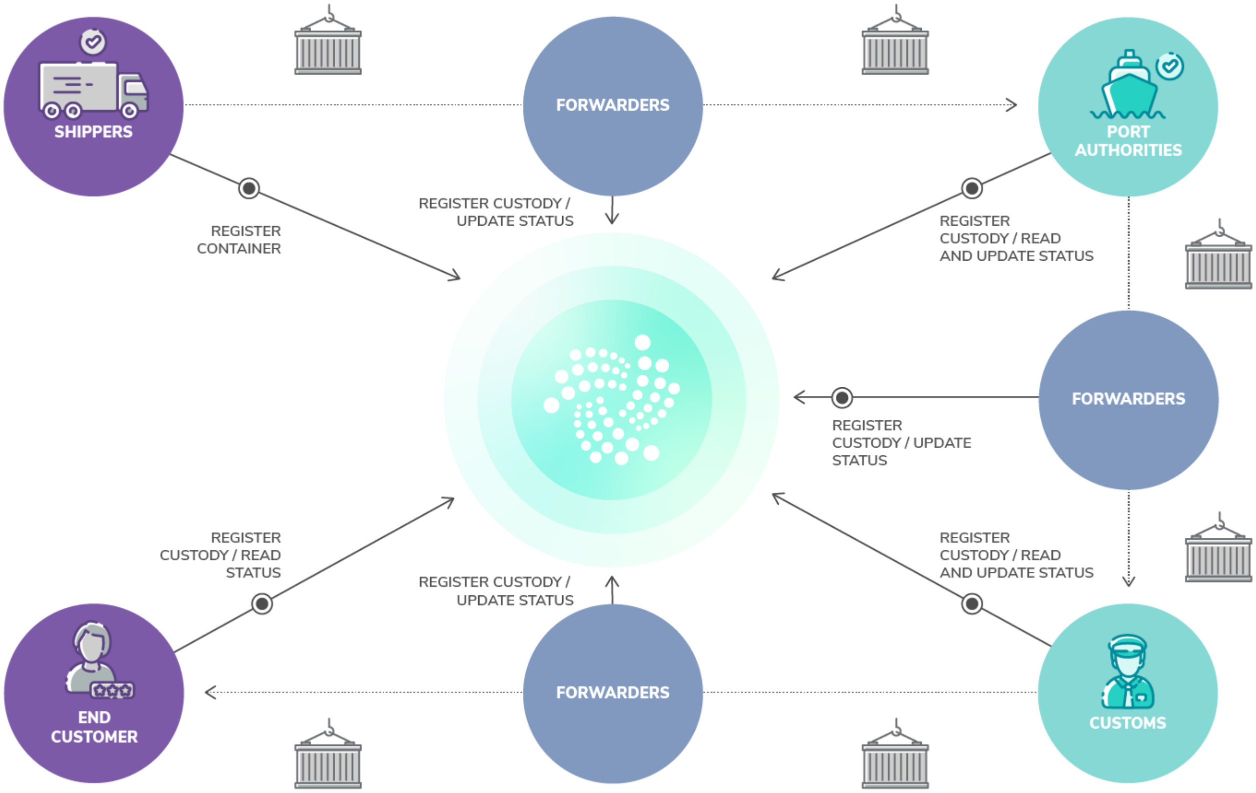

SHIPPERS
FORWARDERS
PORT AUTHORITIES
REGISTER CONTAINER
REGISTER CUSTODY / UPDATE STATUS
REGISTER CUSTODY / READ AND UPDATE STATUS
FORWARDERS
REGISTER CUSTODY / UPDATE STATUS
REGISTER CUSTODY / READ STATUS
REGISTER CUSTODY / UPDATE STATUS
REGISTER CUSTODY / READ AND UPDATE STATUS
END CUSTOMER
FORWARDERS
CUSTOMS

Fig 1: International trade ecosystem

The table below summarizes the different stakeholders and roles considered in our use case:

| **Stakeholders** | **Role** |
|---|---|
| Shippers/Exporters | The shippers are often also the exporter/producer of goods. They load the container with goods and initiate a number of the needed shipping documents for export; then they handover the container and shipping documents to forwarders. Container and shipping documents might be handled by different sets of forwarders. |
| Forwarders | Freight forwarders are agents who coordinate with the other participants in the shipping process on behalf of the importer/exporter. They will coordinate pick-up of container, manage part of the shipping documents |

| | |
|---|---|
| | including transfer to port authorities, customs, shipping liners etc., update container routes. |
| Port Authorities | Port authorities of at least two countries are involved in the shipping process. The port authorities receive the container and documents from the forwarders. They will handle logistics within the port area including moving container to Customs for inspection and loading it onto the vessel. |
| Custom authorities | In any international trade, custom authorities of at least two countries - country of export and country of import - are involved. The customs authorities provide clearance for the goods to leave the country of export and enter the country of import. They need access to shipping documents. |
| End-customers/Importers | They receive container and documents from Forwarders and check container status and transport conditions. |

The table above shows the complexity of the involved ecosystem and the associate number of systems that would require integration in order to allow seamless sharing of the required information.

Instead, this blueprint describes a simple demo platform that allows to share shipment information across any number and type of stakeholders. Such information includes transport conditions of a given container, its location and other monitoring data (e.g., temperature), its chain of custody and other handling events as well as a digital authenticated versions of the associated trade documents, such as Certificate of Origin, Phytosanitary Certificate, packing list etc.

Through the use of the IOTA Tangle (Figure 2) the information associated to the container is shared across all the above actors without requiring any direct integration of proprietary systems. With the use of a distributed ledger, like IOTA, integrity of the shared information is guaranteed, whether this being data or digital documents. This way all the actors handling the container can read and update relevant information without any delay, and so speeding up all the associated operations.

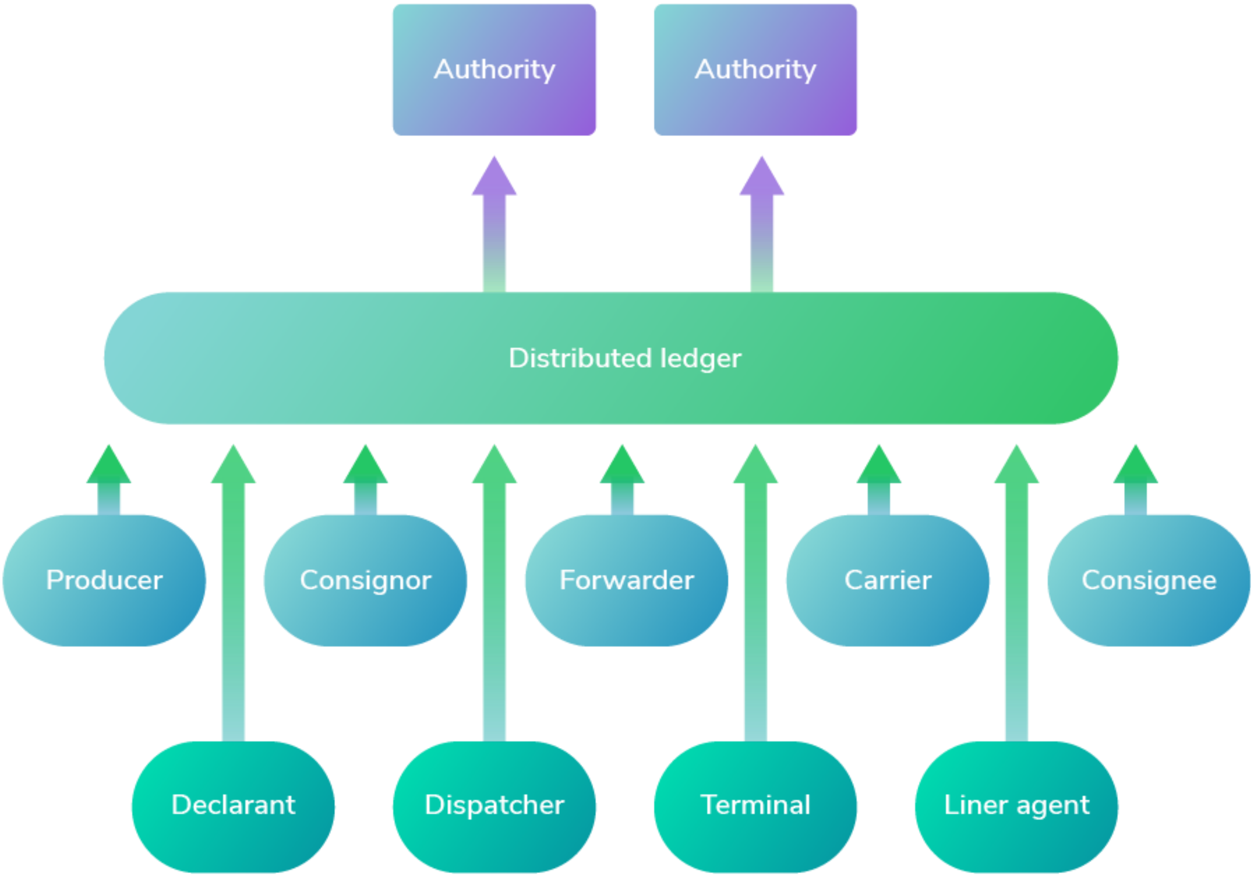
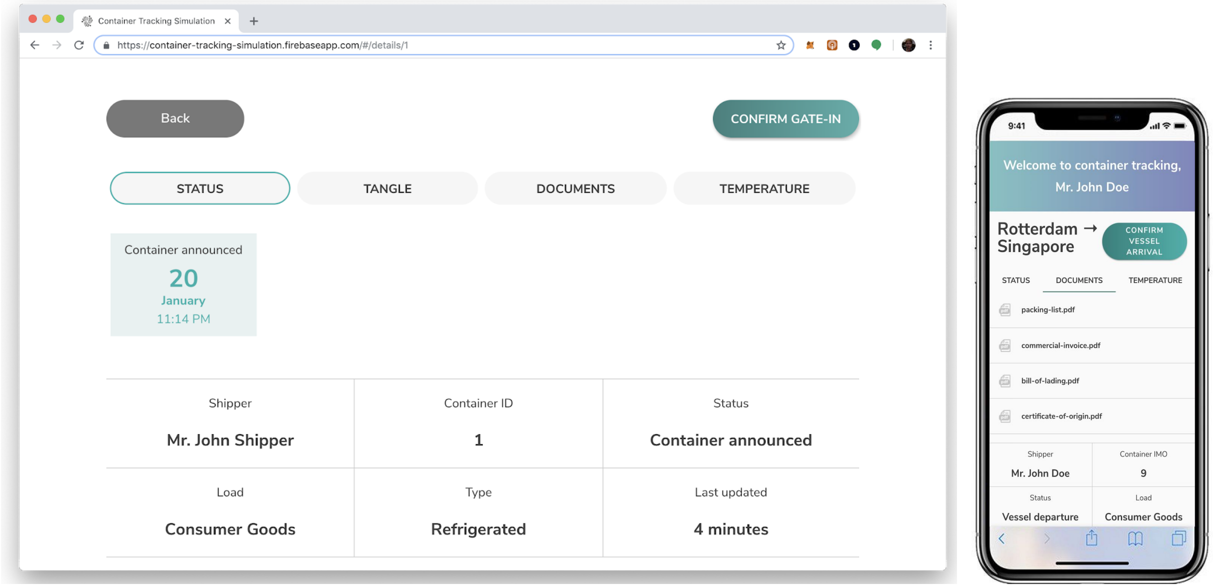


Fig 2: IOTA Trade PoC concept

The solution we describe in this blueprint allows for the following:

- through a web portal or mobile app a *shipper* acquires the given container SSCC number, e.g., by scanning a Barcode, and creates a digital representation of the container, the so called *digital twin*, which includes *container id* and additional information (e.g., container status, load type, temperature, route, position). A digital copy of the shipping documents is also uploaded and a hash of them referenced into the container's digital twin. The updated digital twin is then recorded in an immutable way onto the IOTA Tangle together with the identity of its shipper;
- when the container is handed over to a *forwarder*, through the same portal and QR-code, the new custodian attaches to the container digital twin its identity as well as its location (when available) and updates other relevant information, including specific supply chain handling events. Updated information is recorded onto the Tangle in order to further trace the container journey;
- when the container reaches a *port authority* or a *custom clarence* point, an agent can use the portal or the mobile app to acquire the container identity and, if authorised, to access in real-time all relevant information needed to support her operation, directly from the IOTA Tangle. Following that, an agent can finally issue new events about the container (e.g. Cleared for export) by updating its digital twin. This become immediately visible to all authorised parties. After that, the container is finally delivered to its end-customer, who can verify its whole journey, by retrieving the full digital twin from the IOTA Tangle (from everywhere and at any time).



Container announced
20
January
11:14 PM

| Shipper | Container ID | Status |
|---|---|---|
| Mr. John Shipper | 1 | Container announced |
| **Load** | **Type** | **Last updated** |
| Consumer Goods | Refrigerated | 4 minutes |

Welcome to container tracking, Mr. John Doe

Rotterdam → Singapore

packing-list.pdf
commercial-invoice.pdf
bill-of-lading.pdf
certificate-of-origin.pdf

| Shipper | Container IMO |
|---|---|
| Mr. John Doe | 9 |
| **Status** | **Load** |
| Vessel departure | Consumer Goods |

Fig 3: IOTA Trade PoC

## Business case

International trade is a complex system facing a number of inefficiencies. Figure 4 below shows how international containers shipment of goods is mainly composed of many actors and three flows:

- the physical movement of containers;
- the exchange of data and documents associated to the traded and the transported goods;
- the transfer of any monetary flow associated to the container and the transported goods.

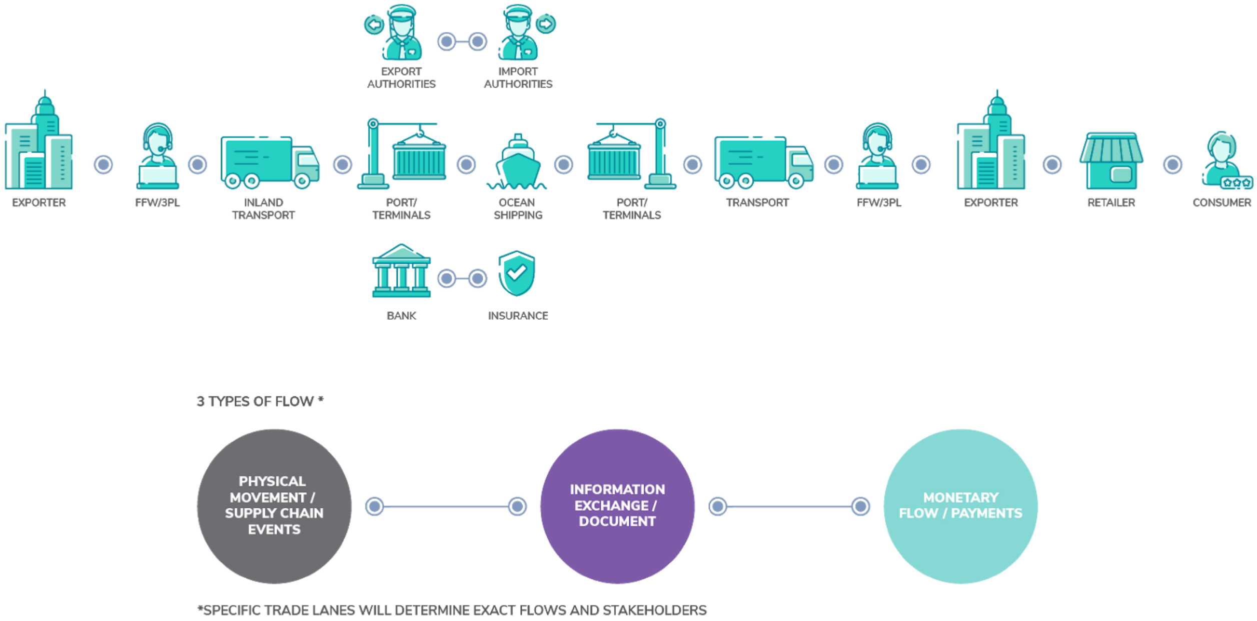


Fig 4: International trade distribution chain

Without considering the third flow, the first two already pose a number of challenges the effect of which is slowing down the speed of international trading. Some of these challenges include:

- Information about transport conditions are collected and stored in siloed systems;

- Full tracking of containers position and chain of custody is not seamlessly visible and in real-time to all the parties handling the container;
- The trading documents and certificates (often paper-based) travel separately from the container and its goods thus many times leading to delay in the clearance procedures, as consequence of document being missing, lost or not timely delivered.

Today's supply chains are driven by actors pushing information to the next actor in the chain (Figure 5) and requesting updates on containers and their content status, approval of shipping documents and payment of required taxes.

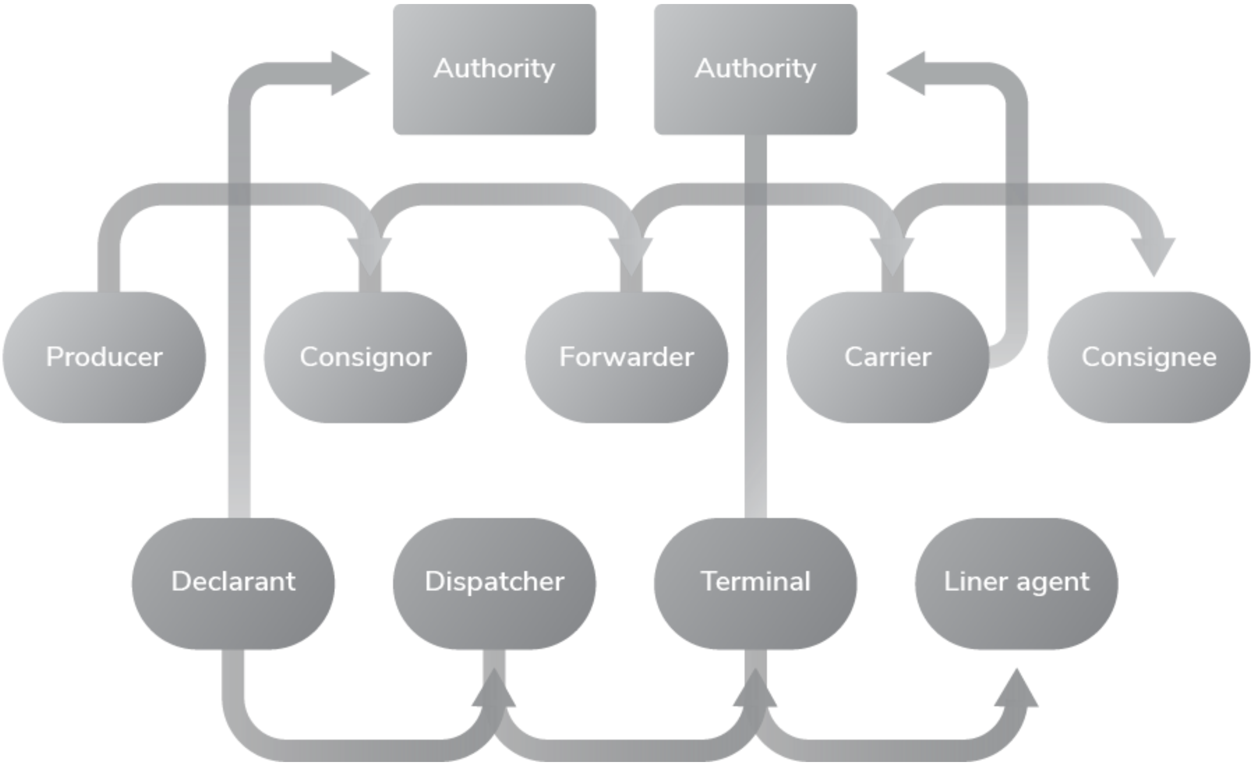


Fig 5: International trading as we know it

Innovation in the international trading has been so far unsuccessful due to the following too established practices:

- Emails, phone calls and paper documents are the daily details of moving goods;
- Information is delivered bilaterally and retyped into new systems with introduction of errors and loss of data integrity and authenticity;
- Multiple data formats are used and often not compatible one with the other.

As result, actors are unable to automatically broadcast/receive notification of events to relevant parties. This generates delay, inefficiencies and loss.

*It is calculated that improving all countries' trade facilitation halfway to global best practice would increase global trade with 15% and global GDP with 4.7 % - before even introducing DLT and Trade Policy 3.0*[1]*.*

[1] http://blogs.lse.ac.uk/businessreview/2018/04/26/disruptive-trade-technologies-will-usher-in-the-internet-of-rules/

Despite of the envisioned benefits, the complexity of the different supply chains systems has so far hindered the seamless sharing of these 3 information flows. This is due partly to technical reasons and greatly to the lack of trust and the risk to share information with unauthorised parties.

Use of distributed ledger technologies, and IOTA in particular, can help to mitigate these risks. The permissionless nature of IOTA Tangle allows for any party to start sharing the required information, with guaranteed integrity. In addition, the use of the 2nd layer MAM[2] protocol allows for fine grain control of information access, despite the distributed nature of the IOTA Tangle. Moreover, using IOTA as trusted data exchange layer, in future scenarios, the use of Token (and IOTA Qubic[3]) could allow to create automated verification of documents and transport conditions and consequently automate moving of associated monetary flows (*trade finance*).

As result, an IOTA powered data exchange layer for trade can deliver the following benefits for each stakeholder category.

For shippers:

- It simplifies paperwork, enables easy way to provide documents and certificates, even when container is already on its way to the destination;
- It enables container position updates and status monitoring;
- It provides overview of chain of custody, handling of goods during shipment;
- It creates an immutable audit trail accessible to refine shipper risk profile and to facilitate their access to services such as trade finance and trade insurance.

For customs clearance :

- It simplifies access to container load information and all related documents and certificates;
- It provides access to shipment information and simplifies direct contact if required;
- It enables government agencies to shift to a Risk-based approach of assessing consignments by enhancing the accuracy and reliability of their risk profiling techniques and tools.

For port authorities and freight forwarders:

- It simplifies access to container route information and estimated time of arrival;
- It provides access to temperature sensor information with optional alerting functionality in case of temperature value rise or power outages;
- It simplifies documentation handling and prevent loss of documents and associated costs.

*It is therefore clear how the development of such trusted data exchange layer can drive innovation and savings for the international trading sector.*

# Architecture

Running an open source project, like any human endeavor, involves uncertainty and trade-offs. We hope the architecture described below helps you to deploy similar systems, but it may include mistakes, and can't address every situation. If you have any questions about your project, we encourage you to do your own research, seek out experts, and discuss with IOTA community.

The presented infrastructure makes use of the IOTA Tangle and IOTA MAM[4].

*The international shipment of containers consists of a chain of events, information and actors involved in the*

[2] A MAM (Masked Authenticated Messaging) is a second layer communication protocol that allows to easily create and read data stream on the IOTA Tangle, guaranteeing integrity of the information and protecting its confidentiality through encryption. More info can be found here: https://blog.iota.org/introducing-masked-authenticated-messaging-e55c1822d50e

[3] Qubic: Quorum-based computation: https://qubic.iota.org/

[4] MAM javascript libraries: https://github.com/iotaledger/mam.client.js

*handling of a given asset (the container and its goods). Because of this, associating each given container shipment to a dedicated MAM channel makes easy to store the different generated information onto the IOTA Tangle as a sequence of MAM messages in the same channel. Using MAM allows for encryption and protection of the shared information. Without using MAM, this could alternatively be done by issuing to the IOTA Tangle independent transactions that store the information generated by each handling procedure of a given container. However, the architecture complexity of reconciling and linking all the information associated to a given container shipment would increase. Hence MAM was chosen as preferred design solution.*

Figure 6 below shows the main architecture components.

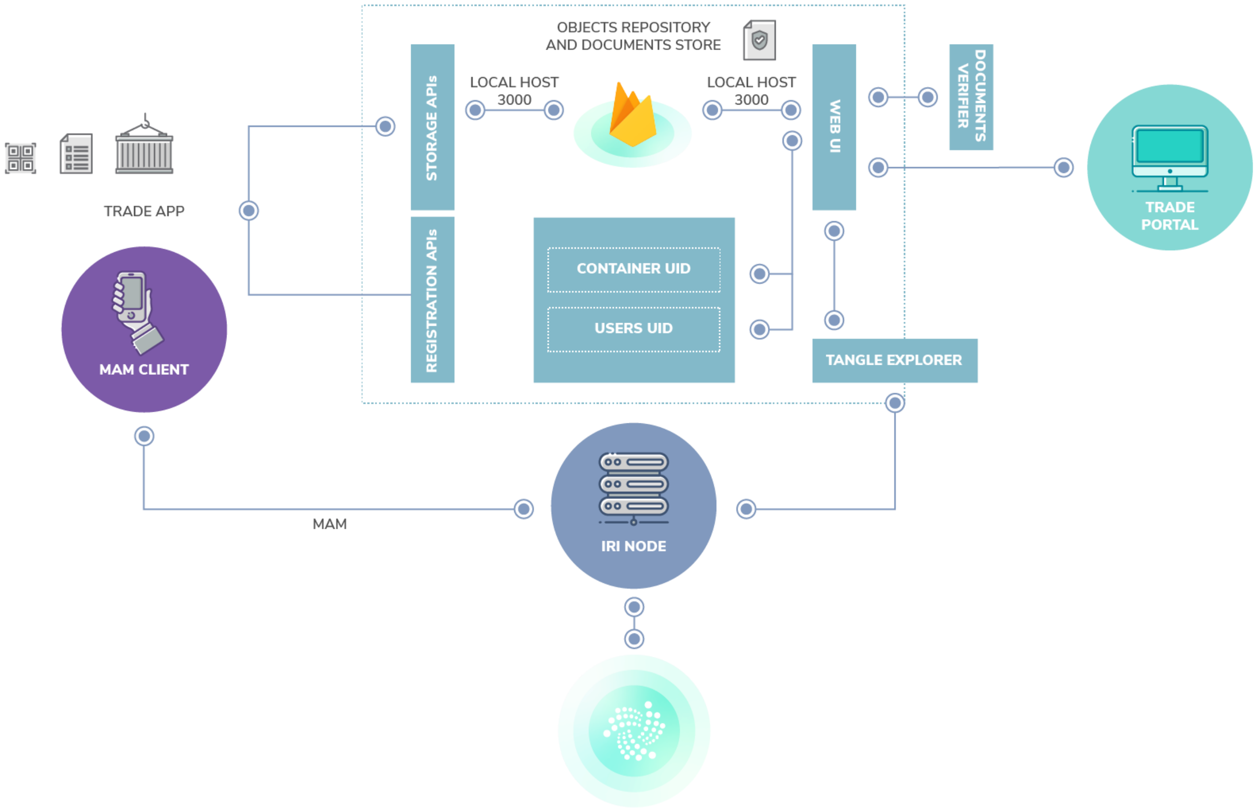


Fig 6: PoC Architecture

Once a new container is first registered by its shipper a new IOTA MAM channel is created. A digital twin for the container is created with the following information: <containerUniqueID, containerOwnerID, containerCustodianID, cargoType, origin, destination, location, temperature, time, status, documents list>. Required information is captured through the Trade PoC app:

- containerUniqueID is captured through Barcode scanning. In future implementation it can be matched against a containerUniqueID server (e.g., GS1 SSCC) for verification purpose;
- information about the containerOwnerID is inserted through the portal. In future implementation it could be fetched from an external source (e.g., a registration server for the use of the app or a self-provided KYC);
- containerCustodianID initially coincide with containerOwnerID;

- location (and temperature) are not implemented but they can optionally be acquired by a dedicated beacon[5] installed into the container;
- time is acquired by the mobile phone or an installed beacon;
- status could corresponds to standard EPCIS Supply Chains events. For this PoC we use a set of predefined standard statuses. Initial status it set to *Container Announced*;
- documents list contains external URLs to relevant shipping documents alongside with their computed hash.

The information is stored to the IOTA Tangle using the javascript MAM client library. This can either be embedded into the app or be implemented through an external server (MAM Server, not shown here), to which the app exchange information using secure HTTPS REST APIs.

> Tip1: A new container shipment is created and announced only once. While the container custody and other tracking information change over time, it is meaningful to use a MAM channel for aggregating information related to a given shipment. This allows to easily link all the different information to the right container and its shipment and to protect access to the relevant information by encrypting each MAM message belonging to a specific channel.

After creation of each MAM channel, a central back-end Object Repository is populated. The Object Repository is implemented as Firebase NoSQL database and deployed using port 3000. Storage REST APIs are provided to populate and update the Firebase DB with information related to the MAM channel associated to a given containerUniqueID. Information stored in the Object Repository includes the root address of the channel, e.g., where this can be accessed on the IOTA Tangle and the cryptographic key needed for decrypting the information stored in the channel (named *side keys*), in case restricted MAM channels are used. The following tuple is created and stored in the Object Repository: <containerUniqueID, channelRoot, channelSideKey>.

The Object repository is either populated by the app or the MAM Server, according to the implemented architecture. Access to the Object Repository is managed by the given container shipper, thus guaranteeing control on who can access and modify the information chain associated to a given container shipment (e.g. by adding new MAM messages).

> Tip2: In a real deployment, it is envisioned that all involved actors will extend their existing ERP management system with capability to store the required information for access to the IOTA Tangle and MAM channel, instead of creating an external central Object Repository.

For a given shipment, when the container changes custodian, information about the new custodian is appended to the existing MAM channel. Additionally location and temperature of the container can also be updated, by the new custodian (or automatically by any beacon installed in the container). For that, a new MAM message, with updated digital twin information, is attached to the existing channel. The following information is updated and stored onto the Tangle: <containerCustodianID, location, temperature, time, status>.

In order to achieve this the mobile app (or the beacon) needs to access, either directly or through the MAM Server the information related to the root of the MAM channel associated to the given container (e.g. where the given channel is stored onto the Tangle). This information is fetched from the Object Repository, by using as primary key the containerUniqueID, which is obtained from the Barcode scanning, manually inserted (or preloaded into the beacon). The following two functions:

```
createItem(eventBody, channel, secretKey, userId);
```

```
updateItem(eventBody, mam, newItemData, user);
```

have been implemented in order to respectively access and update existing MAM channel information (e.g. adding

[5] A wireless sensing unit installed inside the container, e.g. a RuuviTag https://lab.ruuvi.com/iota/ or a Bosch XDK https://www.bosch-connectivity.com/newsroom/blog/xdk2mam/

new messages to update the stored digital twin). Information is then attached to the correct MAM channel and stored immutably onto the IOTA Tangle.

In case of new documents upload or update of existing ones, every time a document is saved by one of the actors in any Document Storage (Figure 6 shows only one for simplicity), its metadata including size, last change date and calculated hash checksum are stored in the IOTA Tangle as part of the digital twin associated to the container. In case of update of existing documents, the original copy hash checksum is retrieved and a new one calculated in real time. If there are differences with the stored values, a Documents Verifier (implemented in the web or mobile app) will send an alarm to the current actor and indicate that documents' content is no longer integral and has been changed.

A Web UI (WUI) written in React implements APIs to access to the MAM explorer and to retrieve information, e.g. container custodian, location, temperature and lists of associated documents and events. Information on the Tangle is retrieved by accessing the required channel root address obtained from the Object Repository.

With the same GUI a list of document hashes associated to a given container can also be retrieved. Documents that have been altered from their initial version are flagged red.

The sequence diagram below recaps all the steps needed to update, track and check a container shipment.

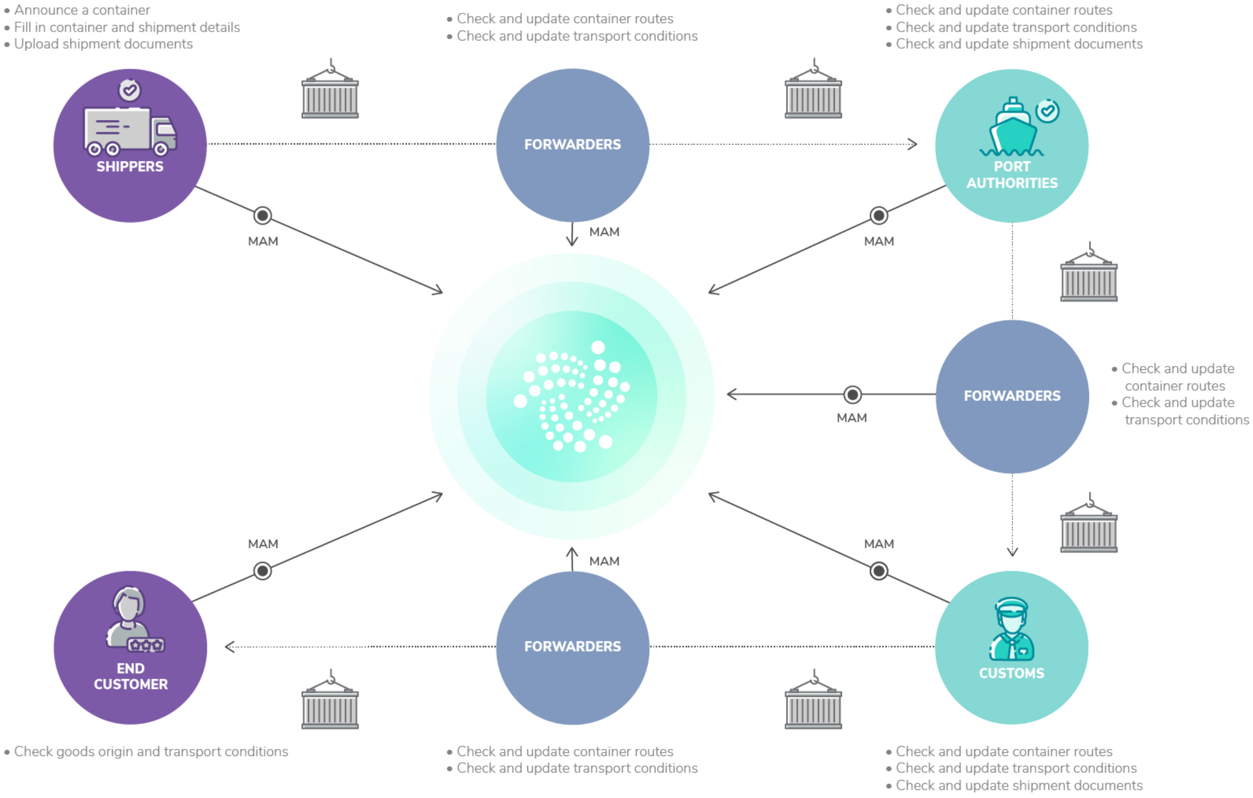

• Announce a container
• Fill in container and shipment details
• Upload shipment documents
• Check and update container routes
• Check and update transport conditions
• Check and update container routes
• Check and update transport conditions
• Check and update shipment documents
SHIPPERS
FORWARDERS
PORT AUTHORITIES
MAM
MAM
MAM
FORWARDERS
• Check and update container routes
• Check and update transport conditions
MAM
MAM
MAM
MAM
END CUSTOMER
FORWARDERS
CUSTOMS
• Check goods origin and transport conditions
• Check and update container routes
• Check and update transport conditions
• Check and update container routes
• Check and update transport conditions
• Check and update shipment documents

Fig 7: PoC Sequence Diagram (actors and actions available)

The communication diagram below shows the different messages exchanged across the architecture components presented above.

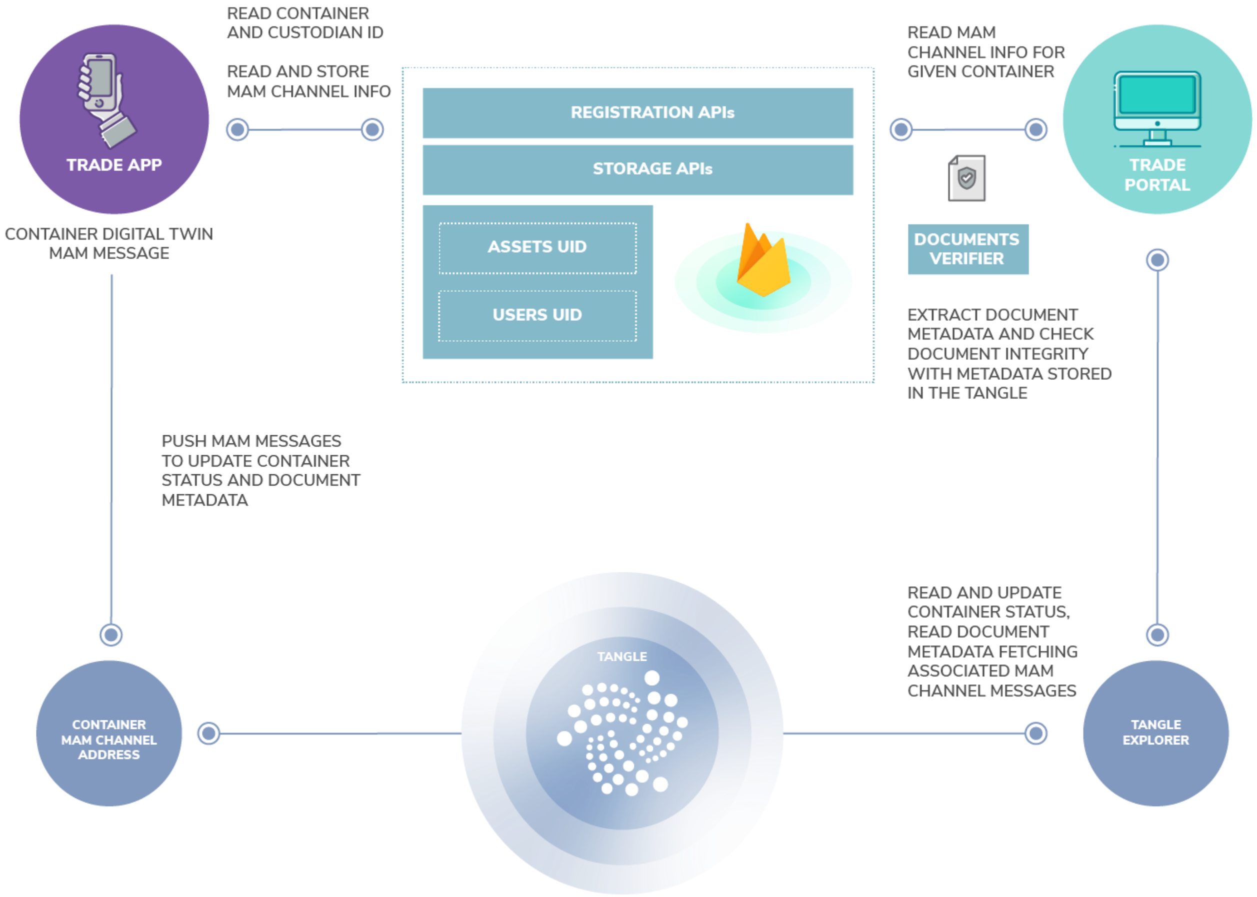

READ CONTAINER AND CUSTODIAN ID
READ AND STORE MAM CHANNEL INFO
TRADE APP
CONTAINER DIGITAL TWIN MAM MESSAGE
REGISTRATION APIs
STORAGE APIs
ASSETS UID
USERS UID
READ MAM CHANNEL INFO FOR GIVEN CONTAINER
TRADE PORTAL
DOCUMENTS VERIFIER
EXTRACT DOCUMENT METADATA AND CHECK DOCUMENT INTEGRITY WITH METADATA STORED IN THE TANGLE
PUSH MAM MESSAGES TO UPDATE CONTAINER STATUS AND DOCUMENT METADATA
READ AND UPDATE CONTAINER STATUS, READ DOCUMENT METADATA FETCHING ASSOCIATED MAM CHANNEL MESSAGES
TANGLE
CONTAINER MAM CHANNEL ADDRESS
TANGLE EXPLORER

Fig 8: PoC Communication Diagram (information flow inside the platform)

Details on the different components implementation is provided below, alongside with code snippets.

## Data model/digital twin

A returnable asset digital twin contains the following:

```
{       "data": [{
                "containerUniqueID": "number",
                "containerOwnerID": "string",
                "containerUserID": "number",
                "location": "string",
                "temperature": "number",
                "time": "date",
                 "documents": [ { "link": "string", "hash": "string"} ],
                "status": "string"
        }]
}
```

## IOTA building blocks

The Trade PoC portal will be responsible of creating and updating container digital twins as MAM channels and messages in order to allow tracking.

```
import Mam from 'mam.client.js';
import { isEmpty, uniqBy, pick, find, last } from 'lodash';
import { asciiToTrytes, trytesToAscii } from '@iota/converter'
import { createItem, updateItem } from './firebase';
import config from '../config.json';
```

To do that, first of all, we need to import the `mam.client.js` library.

Then before creating a MAM channels, we need to select the current IOTA network where transactions will be stored (provider). This could be the main IOTA Network or any dev network, such as: `https://testnet140.tangle.works`

```
// Initialise MAM State with IOTA provider
let mamState = Mam.init(config.provider);
```

For each new assets acquired, we need first to create the returnable asset digital twin (`createItemChannel`).

```
// create the MAM message payload
export const createItemChannel = (project, itemId, request, userId) => {
  const promise = new Promise(async (resolve, reject) => {
    try {
      const secretKey = generateSeed(20);
      const eventBody = {};
      project.firebaseFields.forEach(field => (eventBody[field] = request[field]));
      eventBody.itemId = itemId;
      eventBody.timestamp = Date.now();

      const messageBody = {
        ...request,
        ...eventBody,
        temperature: null,
        position: null,
        lastPositionIndex: 0,
        documents: [],
      };
```

Before setting up the channel, it is recommended to set the channel mode to *'restricted'*. This allows to encrypt the payload (e.g., the information contained in the digital twin) of each MAM message associated to that channel and to guarantee access only to selected parties (`Mam.changeMode()`).

```
// create a new restricted channel
const createNewChannel = async (payload, secretKey) => {
  // Set channel mode for default state
  const defaultMamState = Mam.changeMode(mamState, 'restricted', secretKey);
  updateMamState(defaultMamState);
  const mamData = await publish(payload);
  return mamData;
};
```

We can then publish the information to the IOTA Tangle (`Mam.attach()`). Remember that IOTA uses Trytes so our MAM payload needs to be converted before sending it to the Tangle (`asciiToTrytes(JSON.stringify(data))`) and to create a MAM message (Mam.create()).

```
// store new messages for each new asset and for each change of custody
// Publish to tangle
const publish = async data => {
  try {
    // Create MAM Payload - STRING OF TRYTES
    const trytes = asciiToTrytes(JSON.stringify(data));
    const message = Mam.create(mamState, trytes);

    // Save new mamState
    updateMamState(message.state);

    // Attach the payload.
    await Mam.attach(message.payload, message.address);

    return { root: message.root, state: message.state };
  } catch (error) {
    console.log('MAM publish error', error);
    return null;
  }
};
```

Once a new MAM channel is created or an existing one is updated, we need to update the object repository. This can be done through the `createItem()` and `updateItem()` functions introduced above and described below.

```
export const createItem = (eventBody, channel, secretKey, userId) => {
  // Create item reference
  const itemsRef = getItemReference(eventBody.itemId);
  appendItemToNewUser(userId, eventBody.itemId);

  itemsRef.set({
    ...eventBody,
    mam: {
      root: channel.root,
      seed: channel.state.seed,
      next: channel.state.channel.next_root,
      start: channel.state.channel.start,
      secretKey,
    },
  });
};
```

In the `updateItem()` function below, first the Firebase Object Repository is searched for an existing asset, through its itemId and subsequently information are updated with the new MAM channel or message details.

```
export const updateItem = (eventBody, mam, newItemData, user) => {
  // Create reference
  const itemsRef = getItemReference(eventBody.itemId);

  itemsRef.update({
    ...eventBody,
    mam: {
      root: mam.root,
      secretKey: mam.secretKey,
      seed: newItemData.state.seed,
      next: newItemData.state.channel.next_root,
      start: newItemData.state.channel.start,
    },
  });
};
```

Finally, when a new document is uploaded its integrity is validated. First the original document information are retrieved from the IOTA Tangle (`fetchItem()`).

```
const promise = new Promise(async (resolve, reject) => {
      try {
        const itemEvent = await fetchItem(
          item.mam.root,
          item.mam.secretKey,
          this.storeItemCallback,
          this.setStateCalback
        );

        await validateIntegrity(itemEvent);
        this.setState({ showLoader: false, fetchComplete: true });
        return resolve();
      } catch (error) {
        this.setState({ showLoader: false });
        return reject(this.notifyError(`Error loading ${trackingUnit} data`));
      }
    });
```

The `ValidateIntegrity()` function compares immutable information stored in the IOTA Tangle with the one extracted from the new uploaded document.

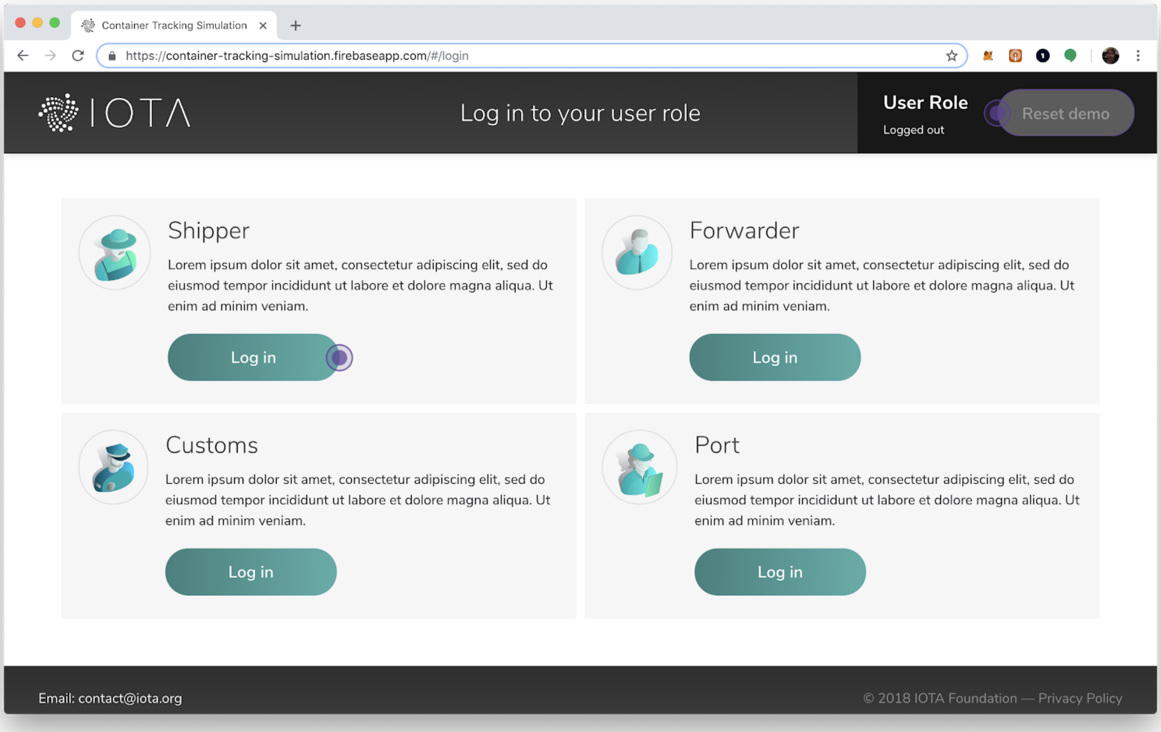
```
export const validateIntegrity = async itemEvent => {
  const promises = [];

  itemEvent.documents.forEach(document => {
    const promise = new Promise((resolve, reject) => {
      try {
        axios
          .get(document.downloadURL, { responseType: 'blob' })
          .then(response => {
            const reader = new FileReader();
            reader.readAsArrayBuffer(response.data);
            reader.onload = () => {
              const arrayBuffer = reader.result;
              const sha256Hash = sha256(arrayBuffer);
              document.hashMatch = sha256Hash === document.sha256Hash;
              document.sizeMatch = response.data.size === document.size;
              resolve();
            };
          })
          .catch(error => {
            reject(error);
          });
      } catch (error) {
        return reject(error);
      }
    });
    promises.push(promise);
  });
```

# Deployment and Testing

The section below describes how this blueprint can be tested, deployed or modified (open source repositories are available).

A web demo of the Trade PoC can be accessed here: [https://container-tracking-simulation.firebaseapp.com/#/login](https://container-tracking-simulation.firebaseapp.com/#/login)



Container Tracking Simulation

https://container-tracking-simulation.firebaseapp.com/#/login

IOTA — Log in to your user role — User Role: Logged out — Reset demo

**Shipper**

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam.

Log in

**Forwarder**

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam.

Log in

**Customs**

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam.

Log in

**Port**

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam.

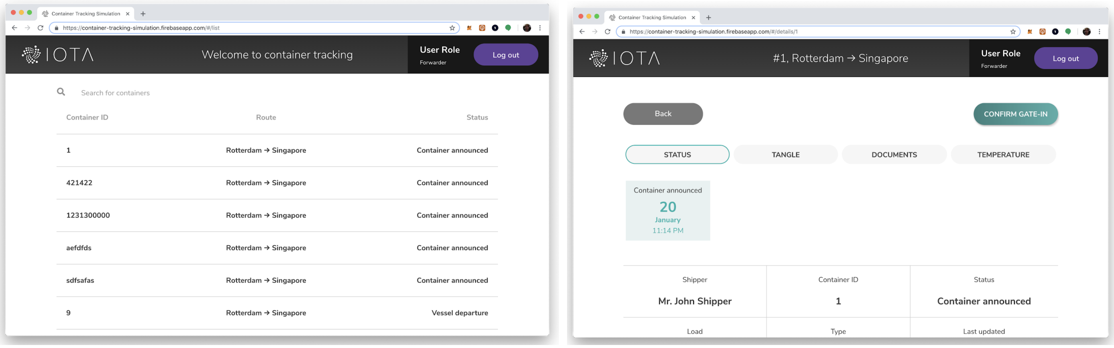
Log in

Email: contact@iota.org

© 2018 IOTA Foundation — Privacy Policy

Fig 9: Login screen for the Trade  PoC

The current implementation of the PoC does not require password. In order to use the service, users can select any available role and start to explore the tracking information related to a given asset. In the following screen, it is possible to browse the information related to a given container, location (and in this case, temperature and attached documents and their integrity).



Container Tracking Simulation

https://container-tracking-simulation.firebaseapp.com/#/list

IOTA — Welcome to container tracking — User Role: Forwarder — Log out

Search for containers

| Container ID | Route | Status |
|---|---|---|
| 1 | Rotterdam → Singapore | Container announced |
| 421422 | Rotterdam → Singapore | Container announced |
| 1231300000 | Rotterdam → Singapore | Container announced |
| aefdfds | Rotterdam → Singapore | Container announced |
| sdfsafas | Rotterdam → Singapore | Container announced |
| 9 | Rotterdam → Singapore | Vessel departure |

Container Tracking Simulation

https://container-tracking-simulation.firebaseapp.com/#/details/1

IOTA — #1, Rotterdam → Singapore — User Role: Forwarder — Log out

Back — CONFIRM GATE-IN

STATUS — TANGLE — DOCUMENTS — TEMPERATURE

Container announced: 20 January 11:14 PM

| Shipper | Container ID | Status |
|---|---|---|
| Mr. John Shipper | 1 | Container announced |
| Load | Type | Last updated |

Fig 10: Container list (left) and container managements screen (right)

It is also possible to create new assets and generate new supply chains events.

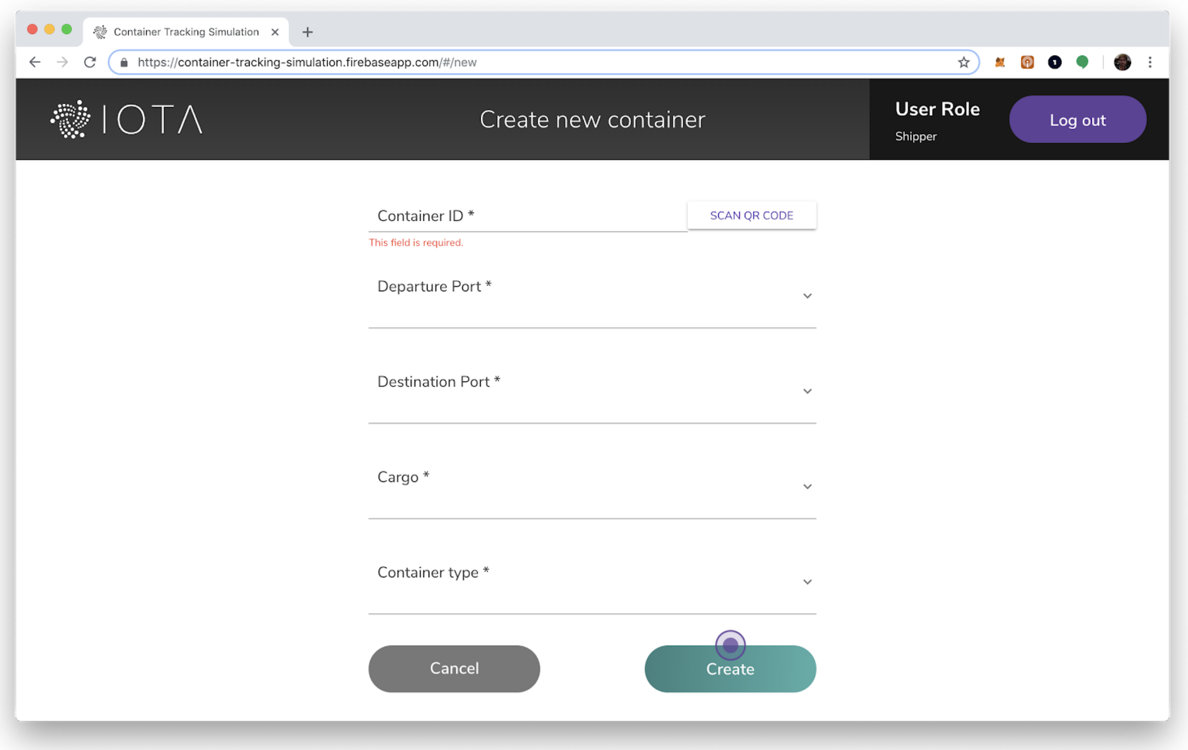


Fig 11: New container creation screen

In a real deployment scenario this frontend portal, passwords are used to regulate access to the ObjectRepository and consequently to the information stored on the Tangle, allowing to manage role access control to each portion of the stored information.

The Demo software is based on the IF open-source IOTA library written in JavaScript. Documentation and usage guide are provided. https://github.com/iotaledger/mam.client.js This library exposes functions for creating a new MAM channel, submitting a new event to the existing channel and retrieving data from the channel. The demo is running on IOTA testnet.

More details and re-usable code can be found here: https://github.com/iotaledger/trade-poc

## Prerequisites

Programming: Javascript and React and basic knowledge of DBMS

For how-to deploy a Firebase server for the required PoC backend functionalities, please read here: https://firebase.google.com/

For how-to connect to an IOTA node, and sending transaction to the IOTA network using IRI, please read here: https://docs.iota.org/iri

For how-to to create MAM Channel and messages, using the IOTA MAM JS library, please read here: https://github.com/iotaledger/mam.client.js

## How to use this blueprint

In order to reproduce this PoC there is no requirement to deploy dedicated HW.

To deploy your track and trace web app, please follow instructions here:

https://github.com/iotaledger/trade-poc/blob/master/firebase_functions/README.md

To deploy your backend server (Object Repository) built in Firebase, please follow instructions here:

https://github.com/iotaledger/trade-poc/blob/master/README.md

*To learn more and to discuss this and similar PoC, please contact Jens* Munch Lund-Nielsen (jens@iota.org) and Michele Nati (michele@iota.org)

# Appendix

## List of Abbreviations

| | | |
|---|---|---|
| IF | IOTA Foundation | |
| HW | Hardware | |
| SW | Software | |
| JS | Javascript | |
| POC | Proof of Concept | |
| DBMS | Database management systems | |
| IRI | IOTA Reference Implementation | the SW written in JAVA that allows users to become part of the [IOTA] network as both a transaction relay and network information provider through the easy-to-use [API]. |
| MAM | Masked Authenticated Messaging | a second layer data communication protocol which adds functionality to publish and control access to an encrypted data stream, over the Tangle |

## Additional Resources

- IRI Repository - https://github.com/iotaledger/iri
- MAM eloquently explained - https://blog.iota.org/introducing-masked-authenticated-messaging-e55c1822d50e
- MAM Source code Repository - https://github.com/iotaledger/mam.client.js
- iota.js Repository - https://github.com/iotaledger/iota.js