Zebra Savanna-IOTA: Track and Trace Ledger .- 101 Tutorial and HowTo

Introduction

Automatic Physical Item Identification is one important aspect when it comes to decentralised trade and supply chains leveraging Distributed Ledger Technologies (a.k.a. blockchains). In this tutorial we are going to provide an introduction to different automatic identification technologies and how they can be seamlessly integrated with an IOTA Ledger (the Tangle) through simple but yet powerful REST APIs. The *Track and Trace Ledger API* is offered as a service (currently in a sandbox-preview version mode) by Zebra Technologies and the IOTA Foundation, through the *Zebra Savanna Data Services* platform. Actually, such APIs are a thin wrapper on top of the **IOTA Streams** technology which enables *Masked Authenticated Messaging* (MAM), a second layer DLT protocol.

The tutorial starts with an introduction to Automatic Identification Technologies, DLTs and IOTA. Then, some of the potential applications of those technologies are briefly discussed. Finally, a programming tutorial on the *Track and Trace Ledger API* is given, together with an introduction to the usage of IOTA APIs and Web-based explorer tools.

Automatic Identification Technologies

Overview

The problem of Automatic Identification and Data Capture (AIDC) has been tackled since the 1970s, with the introduction of linear barcodes. The final aim is to identify (i.e. the recognition of one entity versus another) and capture data (including but not limited to a Digital ID) about trade items facilitating tracking, processing, and inventory processes. For instance, in the retail domain, at the supermarket (or any other point-of-sale) the cashier checks out consumer goods by just scanning a linear barcode printed within the item's label. AIDC technologies allow for greater safety, reliability, speed and efficiency of supply chains.

AIDC technologies include different variants of barcodes, QR Codes, smart cards, biometrics, RFID or even machine learning / deep learning techniques. In this first tutorial we will be focusing on barcodes and RFID.

There are two different concerns to be addressed in AIDC:

- **Identification Scheme** i.e. representation scheme used for ID data. For instance if the identification scheme is numeric, it should define how many digits are used in total and what is the meaning of each digit or group of digits. Error check digits might also be present.
- **Data Carrier**. A Data Carrier is a means to represent AIDC data in a machine readable form. RFID is a data carrier technology based on electronic tags whereas a barcode is a data carrier based on printed symbols.

Barcodes

A number of national and global standardisation bodies have developed barcode technical standards, namely the JTC1 subcommittee of ISO/IEC. Those standards are complemented and extended with others that have to do with global business communication such as GS-1.

In addition to the identification scheme used, the symbology is an important concern when it comes to barcode usage. **Symbology** determines how ID data is encoded, so that it is **machine-readable**. For instance, linear barcode symbologies use some pattern of adjacent, varying width, parallel, rectangular dark bars and pale spaces.

The combination of a symbology together with an identification scheme yields different kinds of barcodes. For instance, the *EAN/UPC Composite* symbology family includes, among others, the **EAN-13** barcode (that can carry GTIN-13 identifiers) and the **UPC-A** barcode (that can carry GTIN-12 identifiers).

EAN-13 barcodes are those which are normally found in consumer goods and its identification scheme is composed by the following elements, as specified by the GS-1 Global Trade Identification Number 13 (GTIN-13) standard:

- A company prefix composed by the country of origin plus a manufacturer code (assigned by the country authority). This can take 7 digits or more.

- The next digits, up to the 12th, encode the product code. The product code is self-assigned by the manufacturer.
- The last digit (the 13th) is a control digit used to verify that a barcode has been scanned correctly.

This GS-1 search service allows you to decode some barcodes.

Sometimes barcodes include (printed) a human readable interpretation i.e. characters, such as letters and numbers. They are there just in case they need to be read and manipulated by humans.

Barcode decoding is realised through specialised electronic devices (*scanners*) that generate identification events to be consumed by an application. A scanner has to provide to the application the symbology used, the identification scheme (i.e what kind of barcode has been identified) and the ID data. Zebra Technologies is one of the leading providers of scanning technology for professional usage. In addition, the advent of smartphone devices equipped with powerful cameras and CPUs/GPUs capable of executing machine learning algorithms has also brought scanning capabilities to consumers and end users.

RFID

RFID is an AIDC technology that relies on radio frequency (RF) waves. RFID is composed of:

- RFID Tags. They are very tiny electronic devices incorporated into or attached to any kind of object (products, tools, animals, goods, etc.). RFID tags are capable of storing (in a non-volatile memory) and transmitting digital data.
- A specialized device, *RFID Reader* and antennas.

There are different classes of tags suitable to different applications and they can operate at different radio frequencies. They can be summarised as follows:

- *Passive Tags* which are enabled to transmit their data when they are close to a Reader. They are cheaper and smaller which makes them easy to incorporate or attach to items. Reading ranges can vary from centimetres to tens of metres. Storage capacity is low (in the range of hundreds of bytes).
- *Active Tags* which contain a battery that provides the energy to the antenna, which is necessary to send encoded radio waves to the Reader. Active tags are more expensive, have higher capacity (in the range of Kilobytes) and are used to

track high value items that have to be scanned over longer distances (in the range of hundreds of meters).

Tags may either be read-only, having a factory-assigned serial number or may be read/write, where object-specific data can be written into the tag by the system user. “Blank” tags may be written by the end user using an RFID Printer.

RFID is similar to barcode technology in concept but presents several differences that makes it suitable for item level tracking in supply chains:

- RFID does not require a visible tag or label to read its stored data, and as a result, data can be gathered without manual intervention.
- The RFID Reader can automatically receive information from multiple tags. As a consequence, it is possible to capture the information concerning an entire shipment or pallet composed by multiple items.
- The RFID Reader can operate at greater distances and tagged objects can be read in different orientations.
- Tags can store more data than barcodes, i.e. not only an ID but also other characteristics of an item such as, size, weight, production date, etc.

To ensure interoperability, RFID is a technology regulated by different ISO/IEC standards and by EPCGlobal, a GS-1 initiative.

The data stored by an RFID tag usually includes an *Electronic Product Code* (EPC). An EPC is aimed at being a universal identifier (ID) for any physical object. EPCs can have multiple representations, human-readable (for instance as URIs) or machine-readable (binary encoding within the memory of RFID tags). The GS1’s EPC Tag Data Standard (TDS) specifies data formats and encodings for EPCs to be stored on passive RFID (UHF Gen 2) Tags. It is important to note that RFID tags may contain other data besides EPC identifiers (and in some applications may not carry an EPC identifier at all).

DLT Technologies and IOTA

A *Ledger* is an information store that keeps final and definitive (*immutable*) records of transactions. A *Distributed Ledger* is a type of ledger that is shared, replicated, and synchronized in a distributed and decentralized manner. A decentralised system is a distributed system wherein control is distributed among the persons or organizations participating in the operation of the system.

IOTA is an open source DLT that enables sharing of any type of data (including IoT data) guaranteeing traceability of their source, alongside with integrity and immutability of the shared information, and dedicated access management, e.g., who can read what. This is possible using 2nd Layer ledger protocols, such as the Masked Authenticated Messaging, (MAM) supported by the IOTA Streams Framework. These types of transactions are called data (or zero-value) transactions.

In contrast with traditional blockchain-based DLTs, IOTA is based on a Directed Acyclic Graph, the Tangle. This video explains how the IOTA's Tangle works. Here you can find a get started guide intended to IOTA's developers with additional references.

Applications of AIDC Technologies and DLT

AIDC Technologies, such as RFID or barcodes, combined with DLT technologies, such as IOTA, can be enablers of a new generation of *decentralised applications*. The final aim is to utilize the (directly or indirectly) captured information through AIDC to build a digital representation (*Digital Twin*) of physical items and their context (location, ownership, etc.). Such Digital Twin representation can be published on a DLT, a secure, decentralised and trusted database that preserves integrity and acts as the single source of truth. Therefore, the DLT actually allows multiple stakeholders to *share data* in real time across the supply chain (on a B2B or B2C scenario).

One example is the track and trace of different physical items to optimise or to make a process more visible and transparent. See for instance this video from Zebra Technologies and IOTA. A tyre, which has an RFID tag attached, is followed using RFID. Thus, every time the tyre passes through a toll plaza (factory, warehouse, transportation, garage), the tyre movement is recorded by the RFID reader and published to the IOTA Ledger, until it reaches the final car where it will be mounted. Later, the driver (the consumer) can also get access and know all the lifecycle of the tyre being used in her car. Similar traceability processes could be enabled with other products of interest to businesses and consumers, for instance food traceability from the farm to the table.

Another example is paperless trade involving multiple countries and stakeholders. You can watch this video where IOTA Foundation and Trade Mark East Africa are digitising cross border trading using a decentralised system based on IOTA. In this case IOTA and Zebra are using RFID technologies to generate different events that track the evolution of consignments (and their individual items) along the supply chain and global trade stakeholders. The combination of AIDC and DLT allows to increase competitiveness by making trade processes more efficient, reducing delivery disputes and uncertainty.

Zebra Savanna Track and Trace Ledger API Overview

Zebra Savanna is a cloud platform, available to developers, that offers different APIs as a service that can facilitate building advanced cloud applications that exploit AIDC (RFID, barcode, etc.) technologies. Zebra Savanna has been conceived to work seamlessly with Zebra devices, but can also be used within other contexts.

Under its sandbox environment, Zebra Savanna has published a new API, the Track and Trace Ledger API which allows to automatically publish and consume *scan* (barcode) or *read* (RFID) transactions (possibly originated from Zebra devices) to an IOTA Ledger (the Tangle). As a result new applications such as those described above can be easily developed.

Tutorial Prerequisites

In order to execute this tutorial the following prerequisites have to be met:

- Zebra Savanna developer account. You have to go to https://developer.zebra.com and then sign up to obtain a new account.
- Zebra Savanna API Key. After signing up/in You need to go to https://developer.zebra.com/user/me/apps in order to register a new application and obtain an API Key that will have to be used to get access to the API.

TODO: Publish Postman

The tutorial uses cUrl commands throughout, but is also available as Postman documentation. All the IOTA hashes (ids) shown have been ellipsed for the sake of legibility.

The tutorial performs a walkthrough over the different APIs offered: the scan API which allows to publish and consume transactions originating from barcode scan devices and the read API which allows to publish and consume transactions generated from RFID Readers.

Last but not least, it is important to note that the current API implementation sandbox makes use of the IOTA's **Devnet** network. This network is composed of a limited number of nodes, mainly contributed by the IOTA Foundation. Security and confidentiality of transactions issued in this network are the same as per the IOTA Mainnet.

Tutorial Part I : Barcode scan transactions

Create a new barcode scan transaction

Using your API Key a new barcode scan transaction can be published to the Track and Trace Ledger. It is the same operation that would be executed by a scanning device (Zebra scanner, mobile phone, etc.) once a barcode has been scanned. You can observe that the payload includes the identifier of the scanning device (under the key `deviceId`), the type of transaction (`scanTransaction`), the device's generated timestamp, the symbology, the value read (observe that is is an EAN-13 barcode), the location (optional) and other extended information that an application may need (`jsonData` field, which is optional).

Request:

```
curl -i --location --request POST
'https://sandbox-api.zebra.com/v2/ledger/tangle/scan' \
--header 'apikey: <Your API Key>' \
--header 'Content-Type: application/json' \
--data-raw '{
   "symbology": "EAN-13",
   "value": "3700123300014",
   "timestamp": "2020-10-14T16:10:07.652Z",
   "location":{
      "latitude": 44.1,
      "longitude": -8
   },
   "deviceId": "iphone-A-456789",
   "type": "scanTransaction"
}'
```

Response:

```
201 Created

Location: /scan/transaction/JPOPF...
```

After executing the API operation you can observe that a transaction identifier `JPOPF...` has been obtained. Such transaction identifier actually corresponds to the root of a Streams Channel associated to the scanned item. Each ID involved during scan processes is associated with a Channel, so that it is very easy to track what is happening with a particular item. However, thanks to the Masked Authenticated Messaging (MAM) functionality offered by IOTA Streams, the data is stored on the Tangle guaranteeing immutability and confidentiality.

Get the details of a barcode scan transaction

We can obtain the details of our scan transaction by using the identifier obtained in the previous step. Behind the scenes, the API will query the IOTA Tangle and obtain our data. You can observe that the query has provided some extra details about our transaction under the `jsonData` field: A server-side timestamp and a side key. The latter is the side symmetric key (generated automatically by the API implementation) that would allow to decrypt and read the data directly from the Tangle. In fact, you can check that the transaction is actually on the Tangle through the MAM Subscriber Explorer (or by opening a URL in your browser of the form `https://utils.iota.org/mam/JPOPF.../restricted/{yourSideKey}/devnet`).

In addition, using the MAM.js libraries, you could also programmatically get access to the data by using simple Javascript APIs. The usage of MAM.js might facilitate the consumption of Track and Trace Ledger data in different application contexts, for instance, to enable a more efficient refresh of a Web UI.

Request:

```
curl -i --location --request GET
'https://sandbox-api.zebra.com/v2/ledger/tangle/scan/transaction/JPOPF...' \
--header 'apikey: <Your API Key>'
```

Response:

```
{
    "symbology": "EAN-13",
    "value": "3700123300014",
    "timestamp": "2020-10-14T16:10:07.652Z",
    "location": {
        "latitude": 44.1,
        "longitude": -8
    },
    "deviceId": "iphone-A-456789",
    "type": "scanTransaction",
    "jsonData": {
        "timestamp": 1600273876330,
        "sideKey": "LEOXJ..."
    },
    "id": "JPOPF..."
}
```

Later, we could scan the same item, from another location and using another device `Zebra-MC-9200-PA12`, and a similar transaction would be issued. If you are still connected to the MAM Channel Explorer Web page (URL of the form `https://utils.iota.org/mam/JPOPF.../restricted/{yourSideKey}/devnet)` corresponding to your item you will observe that a new transaction will appear.

Obtain a list of the scan transactions associated to an item

You can know what has been happening with an item (i.e. transactions involved) by just supplying the item's barcode value to the API as follows. As a result you will obtain an array list with all the transactions involving such an item.

Below you can observe that both transactions have the same side key associated with, that is due to the fact that all the transactions involving an item are recorded on the same Streams Channel.

Request:

```
curl -i --location --request GET
'https://sandbox-api.zebra.com/v2/ledger/tangle/scan/3700123300014' \
--header 'apikey: <your API Key>'
```

Response:

```
[
    {
        "symbology": "EAN-13",
        "value": "3700123300014",
        "timestamp": "2020-10-14T16:10:07.652Z",
        "location": {
            "latitude": 44.1,
            "longitude": -8
        },
        "deviceId": "iphone-A-456789",
        "type": "scanTransaction",
        "jsonData": {
            "timestamp": 1600273876330,
            "sideKey": "LEOXJ..."
        },
        "id": "JPOPF..."
    },
    {
        "symbology": "EAN-13",
        "value": "3700123300014",
        "timestamp": "2020-10-14T17:11:27.762Z",
        "location": {
            "latitude": 41.6,
            "longitude": -4
        },
        "deviceId": "Zebra-MC-9200-PA12",
        "type": "scanTransaction",
        "jsonData": {
            "timestamp": 1600275780264,
            "sideKey": "LEOXJ..."
        },
        "id": "MB9DC..."
    }
]
```

Obtain a list of the transactions made with a scanning device.

The API is designed so that it is also possible to know all the transactions that have originated from a particular scanning device. For such purpose, a dedicated Stream Channel is also maintained for each scanning device.

You can observe that a device transaction contains a reference to the corresponding transaction at item level `transactionId` field). As it happens with item value ID transactions described above, the information can be accessed directly through the Tangle by using the MAM Subscriber explorer or the MAM.js APIs.

In addition, you can observe that this time the side key is different. The rationale is that the Stream Channel used for storing device transactions is different from the Stream Channel used for storing item transactions (even though they essentially contain the same data).

Request:

```
curl -i --location --request GET
'https://sandbox-api.zebra.com/v2/ledger/tangle/scan/device/Zebra-MC-9200-PA12' \
--header 'apikey: <your API Key>'
```

Response:

```
[
    {
        "transactionId": "MB9DC...",
        "symbology": "EAN-13",
        "value": "3700123300014",
        "timestamp": "2020-10-14T17:11:27.762Z",
        "location": {
            "latitude": 41.6,
            "longitude": -4
        },
        "deviceId": "Zebra-MC-9200-PA12",
        "type": "scanTransaction",
        "jsonData": {
            "timestamp": 1600275780264,
            "sideKey": "9YTQZ..."
        },
        "id": "NQFP..."
    }
]
```

There are other API operations available not covered by this tutorial, for instance, you can list all the scan transactions performed, through the `GET /scan` resource. However, you have to take into account that the number of resulting transactions might be huge and, currently, the sandbox-preview version of the API does not offer paginated payloads. In such cases, where unmanageable payload length is expected, it could be advisable to consume the data directly from the Tangle using the MAM.js library.

Tutorial Part II : RFID Read Transactions

Create a new RFID read transaction

Everytime an RFID reader reads an RFID tag a new transaction to the Tangle is generated as follows.

Request:

```
curl -i --location --request POST
'https://sandbox-api.zebra.com/v2/ledger/tangle/read' \
--header 'apikey: <your API Key>' \
--header 'Content-Type: application/json' \
--data-raw '{
  "deviceId": "MC333R-GI4HG4EU-Vall-1",
  "antennaId": "19495783",
  "peakRssi": -42,
  "epc": "urn:epc:id:ginc:0614141.xyz3311cba",
  "timestamp": "2020-10-09T12:33:59.452Z",
  "type": "readTransaction",
  "location": {
    "latitude": 41.65518,
    "longitude": -4.72372
  }
}
'
```

Response:

The location header contains a reference to the transaction created.

```
201 Created
```

```
Location: /read/transaction/FYUDOTS...
```

You can see that for RFID data the type of transaction is `readTransaction` and it includes the **EPC** just read. Other technical information about the antenna involved, signal and device is also provided. The location where the transaction happened is also sent, in this case as GPS coordinates, but in the future, other location encoding conventions could also be used.

The response obtained is as follows. As it happens with barcode scanning transactions, there is a transaction id which corresponds to the root of the Stream Channel that is being used to record transactions involving the concerned EPC (`urn:epc:id:ginc:0614141.xyz3311cba`).

Now imagine that the tracked item (a consignment in this case) has now reached another location, a customs post, for instance. At that new location its RFID tag is read and another transaction is generated:

Request:

```
curl -i --location --request POST
'https://sandbox-api.zebra.com/v2/ledger/tangle/read' \
--header 'apikey: <your API Key>' \
--header 'Content-Type: application/json' \
--data-raw '{
  "deviceId": "FX9600-Bur-1",
  "antennaId": "78904512",
  "peakRssi": -52,
  "epc": "urn:epc:id:ginc:0614141.xyz3311cba",
  "timestamp": "2020-10-15T06:50:48.801Z",
  "type": "readTransaction",
  "location": {
    "latitude": 42.35022,
    "longitude": -3.67527
  }
}
'
```

Response:

The location header contains a reference to the transaction created.

```
201 Created

Location: /read/transaction/NGBGDWZ...
```

Finally, our initial reader scans another physical item which EPC is `urn:epc:id:sgtin:0614141.012345.Serial`.

Get the details of an RFID read transaction

You can get the details of the read transaction as follows:

Request:

```
curl -i --location --request GET
'https://sandbox-api.zebra.com/v2/ledger/tangle/read/transaction/FYUDOTS...' \
--header 'apikey: <your API Key'
```

Response:

```
{
    "deviceId": "MC333R-GI4HG4EU-Vall-1",
    "antennaId": "19495783",
    "peakRssi": -42,
    "epc": "urn:epc:id:ginc:0614141.xyz3311cba",
    "timestamp": "2020-10-09T12:33:59.452Z",
    "type": "readTransaction",
    "location": {
        "latitude":  41.65518,
        "longitude": -4.72372,
    },
    "jsonData": {
        "timestamp": 1602744088719,
        "sideKey": "IBEPW..."
    },
    "id": "FYUDOTS..."
}
```

Also this information could have been retrieved using the MAM.js library or the MAM Subscriber Explorer (through an URL of the form `https://utils.iota.org/mam/FYUDOTS.../restricted/{yourSideKey}/devnet`).

Obtain a list of the RFID read transactions of an EPC

We can obtain a list of all the read transactions involving an EPC, `urn:epc:id:ginc:0614141.xyz3311cba`, as shown below. You can observe that we get the data of both transactions, and as a result, we are obtaining all the trace that has been followed by our consignment across the trade and supply chain. Similarly as it happens with scan transactions, it can be observed that the rendered side key is the same for both transactions. It is due to the fact that transactions corresponding to the same EPC are linked sequentially to the same Streams Channel, starting from the first transaction, in our example, the one identified by `FYUDOTS...`.

Remember that the same information could have been retrieved through the Tangle using the MAM Explorer (through a URL of the form `https://utils.iota.org/mam/FYUDOTS.../restricted/{yourSideKey}/devnet`) or the MAM.js libraries.

Request:

```
curl -i --location --request GET
'https://sandbox-api.zebra.com/v2/ledger/tangle/read/urn:epc:id:ginc:0614141.xyz3311c
ba' \
--header 'apikey: <your API Key>'
```

Response:

```
[
    {
        "deviceId": "MC333R-GI4HG4EU-Vall-1",
        "antennaId": "19495783",
        "peakRssi": -42,
        "epc": "urn:epc:id:ginc:0614141.xyz3311cba",
        "timestamp": "2020-10-09T12:33:59.452Z",
        "type": "readTransaction",
        "location": {
            "latitude": 41.65518
            "longitude": -4.72372,
         },
         "jsonData": {
            "timestamp": 1602744088719,
            "sideKey": "IBEPW..."
         },
        "id": "FYUDOTS..."
    },
    {
        "deviceId": "FX9600-Bur-1",
        "antennaId": "78904512",
        "peakRssi": -52,
        "epc": "urn:epc:id:ginc:0614141.xyz3311cba",
        "timestamp": "2020-10-15T06:50:48.801Z",
        "type": "readTransaction",
        "location": {
            "latitude": 42.35022,
            "longitude": -3.67527
        },
        "jsonData": {
            "timestamp": 1602744668677,
            "sideKey": "IBEPW..."
         },
        "id": "NGBGDWZ..."
    }
]
```

Obtain a list of the read RFID transactions performed by an RFID Reader

Likewise we could get access to all the read transactions performed by our first RFID Reader, identified as `MC333R-GI4HG4EU-Vall-1`. Remember that each Reader has an independent Stream Channel assigned, so that we can retrieve all its activity (preserving data integrity and authenticity). You can observe that all the transactions we have made so far involving such a reader are listed. You can also observe that there is a pointer (`transactionId`) to the corresponding EPC transaction. The side key involved in this case is different as it corresponds to a different Stream Channel, the one associated to the Reader.

Again you can get access to this data through the MAM Explorer by building a URL of the form `https://utils.iota.org/mam/FYUDOTS.../restricted/{yourSideKey}/devnet`) or through the MAM.js libraries.

Request:

```
curl -i --location --request GET
'https://sandbox-api.zebra.com/v2/ledger/tangle/read/device/MC333R-GI4HG4EU-Vall-1' \
--header 'apikey: <your API Key'
```

Response:

```
[
    {
        "transactionId": "FYUDOTS...",
        "deviceId": "MC333R-GI4HG4EU-Vall-1",
        "antennaId": "19495783",
        "peakRssi": -42,
        "epc": "urn:epc:id:ginc:0614141.xyz3311cba",
        "timestamp": "2020-10-09T12:33:59.452Z",
        "type": "readTransaction",
        "location": {
            "latitude": -4.72372,
            "longitude": 41.65518
        },
        "id": "JQ9SLFB...",
        "jsonData": {
            "timestamp": 1602744090255,
            "sideKey": "EFTYH..."
         }
    },
    {
        "transactionId": "PH9MZ...",
        "deviceId": "MC333R-GI4HG4EU-Vall-1",
        "antennaId": 19495783,
        "peakRssi": "-52",
        "epc": "urn:epc:id:sgtin:0614141.012345.Serial",
        "timestamp": "2020-10-15T07:46:10.542Z"
        "type": "readTransaction",
        "location": {
            "latitude": 41.65518,
            "longitude": -4.72372,
        },
        "id": "XRM9L...",
        "jsonData": {
            "timestamp": 1602748166040,
            "sideKey": "EFTYH..."
         }
    }
]
```

There are other API operations available, not covered by this tutorial, for instance, you can list all the read transactions performed, through the `GET /read` resource. However, you have to take into account that the number of resulting transactions might be huge and, currently, the sandbox-preview version of the API does not offer paginated payloads. In such cases, where unmanageable payload length is expected, it could be advisable to consume the data directly from the Tangle using the MAM.js library.

API Roadmap

The Track and Trace Ledger API (sandbox version) opens up a new world of business opportunities and applications thanks to the combination of IOTA and RFID or barcodes. IOTA Foundation and Zebra technologies intend to continue working on improving and polishing the API by making it enterprise-ready and available on ZebraNet, a IOTA network specifically devoted to Zebra Savanna developers. Some of the upcoming planned features are targeted to improved performance, scalability and robustness. For instance, adding up pagination capabilities, transaction filtering by date and further alignment / harmonization with GS-1 or other Global Trade and Supply Chain industry standards. Last but not least, the upcoming new features and approach of IOTA Chrysalis Phase 2 will bring considerable improvements in how the information is managed and stored, making the system more efficient, performant and scalable.