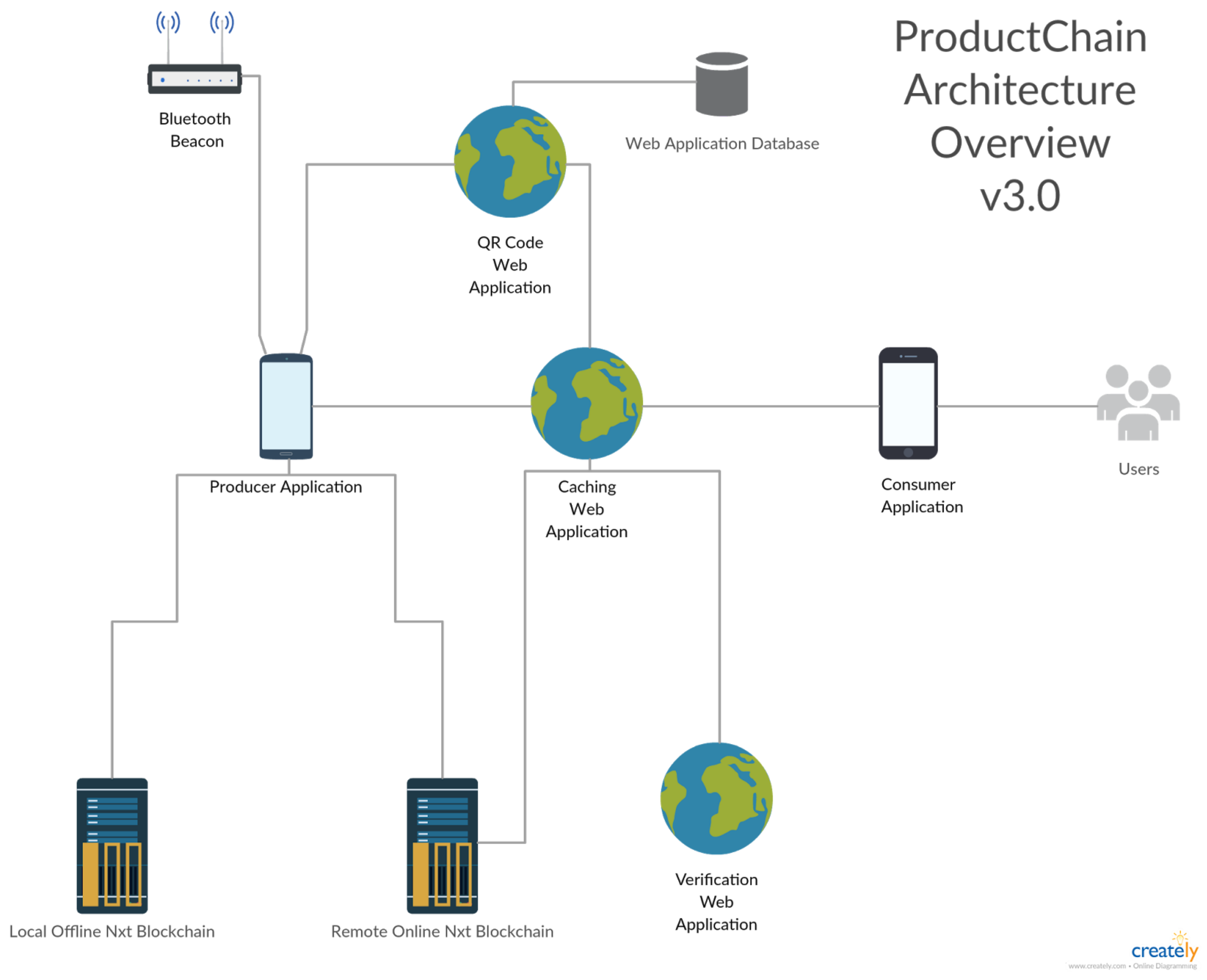
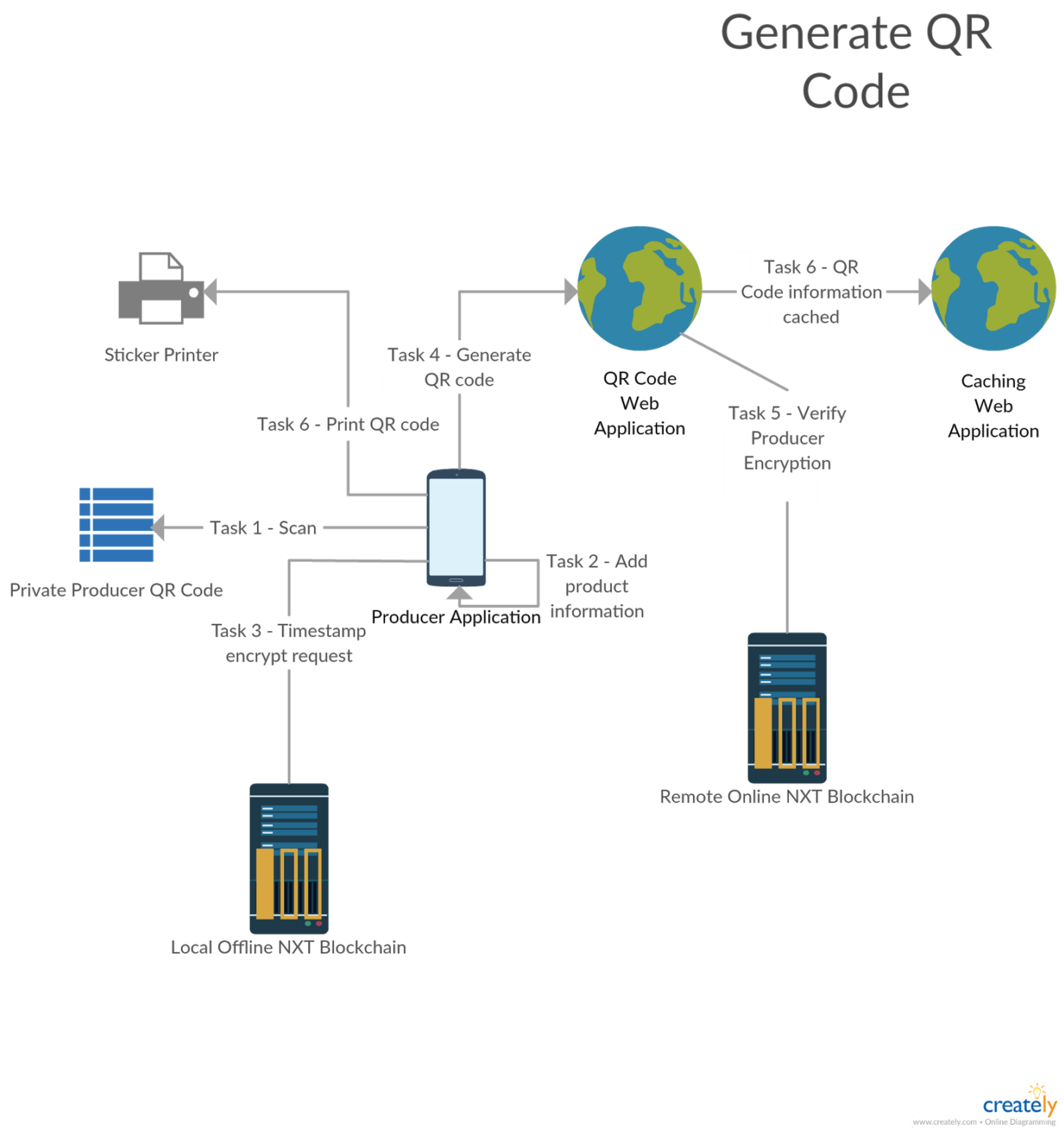
ProductChain Detailed Overview

Architecture Diagram

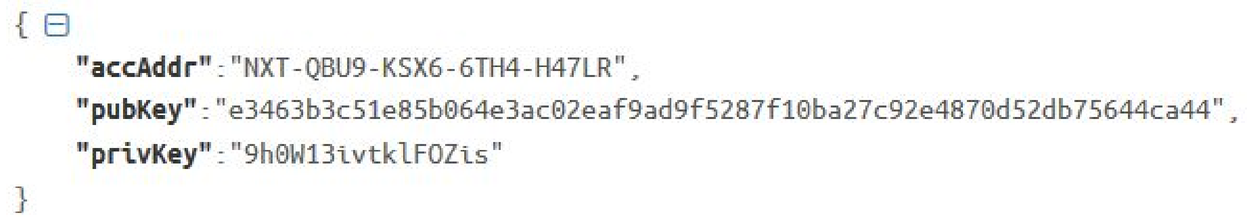

ProductChain Architecture Overview v3.0
Bluetooth Beacon
Web Application Database
QR Code Web Application
Producer Application
Caching Web Application
Consumer Application
Users
Local Offline Nxt Blockchain
Remote Online Nxt Blockchain
Verification Web Application
creately
www.creately.com • Online Diagramming

Generate QR Code


Generate QR Code
Sticker Printer
Task 4 - Generate QR code
QR Code Web Application
Task 6 - QR Code information cached
Caching Web Application
Task 5 - Verify Producer Encryption
Task 6 - Print QR code
Task 1 - Scan
Private Producer QR Code
Task 2 - Add product information
Producer Application
Task 3 - Timestamp encrypt request
Remote Online NXT Blockchain
Local Offline NXT Blockchain
creately
www.creately.com • Online Diagramming

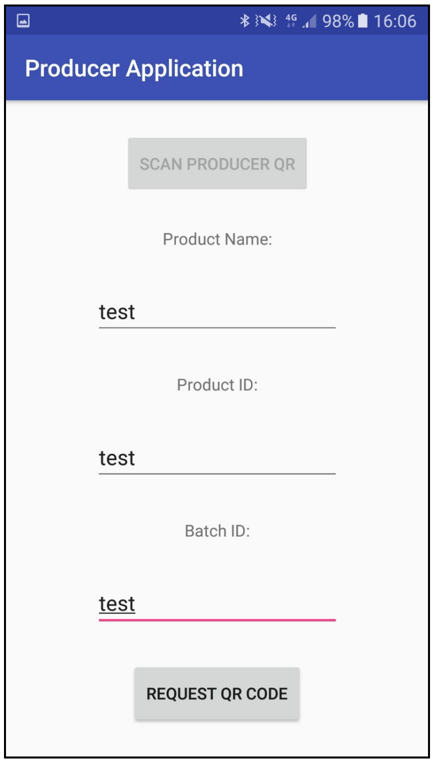
Gather New QR Code Information

The user opens the **Producer Application** then presses the *Request new QR Code* button on the home screen. The User will then press *Scan Producer QR*. This will open the ZXing barcode scanner, that will scan the Producer’s Private QR code. The Producer application will then retrieve the Producer’s NXT account, Private key and Public key from the QR code in json format.



```
{
    "accAddr":"NXT-QBU9-KSX6-6TH4-H47LR",
    "pubKey":"e3463b3c51e85b064e3ac02eaf9ad9f5287f10ba27c92e4870d52db75644ca44",
    "privKey":"9h0W13ivtklFOZis"
}
```

The user then enters the Product Name, Producer ID and Batch ID into the application and presses *Request QR code.*



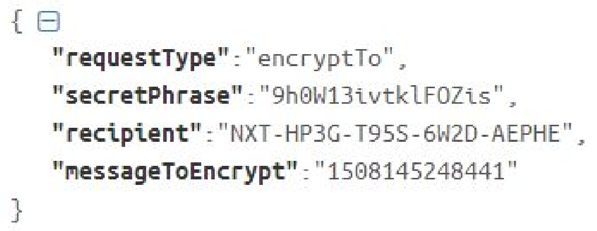
Product Name:
test

Product ID:
test

Batch ID:
test

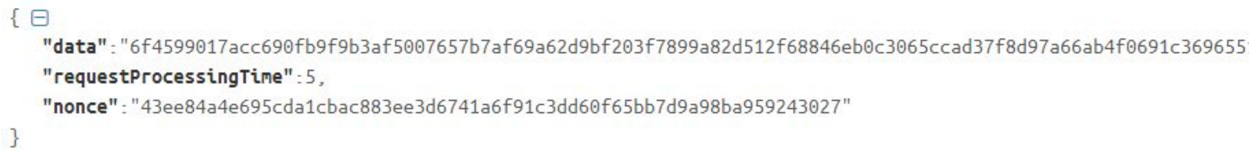
When the *Request QR Code* button is pressed, it gets the current epoch time in milliseconds (messageToEncrypt). This is sent to the **Local Offline NXT Blockchain node** via an *HTTP POST* call along with the producer’s private key (secretPhrase) obtained from the Producer’s QR code and the `PRODUCT_CHAIN_NXT_ADDR` (recipient), which is the NXT account address that controls the administrative features which is a constant in the Producer Application.

Below is a screenshot of the values within the POST request to the offline Nxt node.



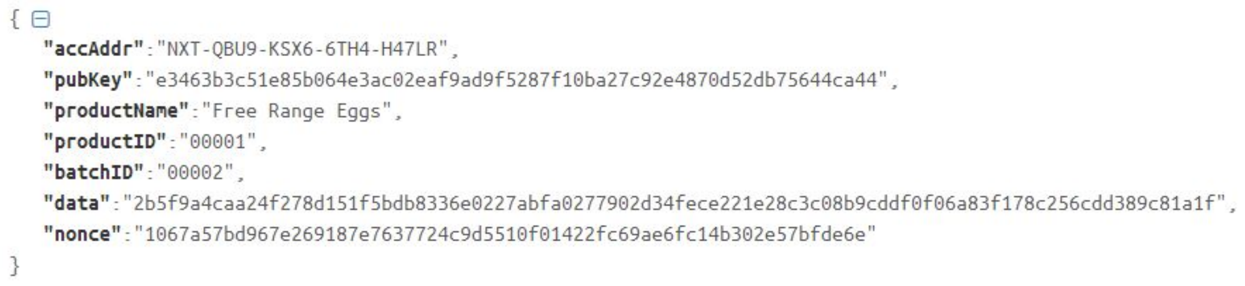
```
{ ⊟
   "requestType":"encryptTo",
   "secretPhrase":"9h0W13ivtklFOZis",
   "recipient":"NXT-HP3G-T95S-6W2D-AEPHE",
   "messageToEncrypt":"1508145248441"
}
```

The offline node then responds with a json object containing the data and nonce fields required after encrypting the timestamp



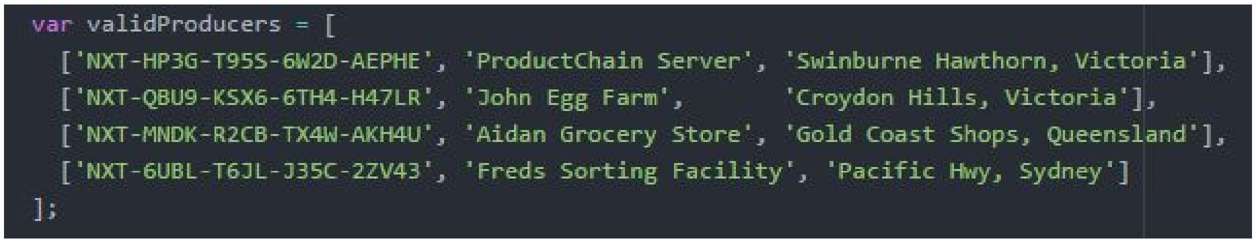
```
{ ⊟
   "data":"6f4599017acc690fb9f9b3af5007657b7af69a62d9bf203f7899a82d512f68846eb0c3065ccad37f8d97a66ab4f0691c369655
   "requestProcessingTime":5,
   "nonce":"43ee84a4e695cda1cbac883ee3d6741a6f91c3dd60f65bb7d9a98ba959243027"
}
```

The **Producer Application** then sends an *HTTP POST* to the QR Code Server containing the information gathered from the QR code, user inputted information about the product as well as the data and nonce received from the Local Offline NXT Blockchain node.



```
{ ⊟
   "accAddr":"NXT-QBU9-KSX6-6TH4-H47LR",
   "pubKey":"e3463b3c51e85b064e3ac02eaf9ad9f5287f10ba27c92e4870d52db75644ca44",
   "productName":"Free Range Eggs",
   "productID":"00001",
   "batchID":"00002",
   "data":"2b5f9a4caa24f278d151f5bdb8336e0227abfa0277902d34fece221e28c3c08b9cddf0f06a83f178c256cdd389c81a1f",
   "nonce":"1067a57bd967e269187e7637724c9d5510f01422fc69ae6fc14b302e57bfde6e"
}
```

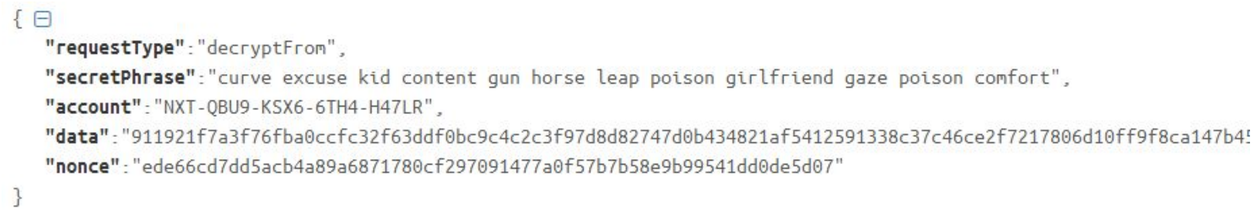
Once this information is received, the QR Code server checks to see if the Nxt address requesting the new QR code is a valid Producer. It does this by checking its database containing a list of valid producers. This list of producers is part of the publicly available information described “Publicly Available Information” documentation.



```
var validProducers = [
  ['NXT-HP3G-T95S-6W2D-AEPHE', 'ProductChain Server', 'Swinburne Hawthorn, Victoria'],
  ['NXT-QBU9-KSX6-6TH4-H47LR', 'John Egg Farm',       'Croydon Hills, Victoria'],
  ['NXT-MNDK-R2CB-TX4W-AKH4U', 'Aidan Grocery Store', 'Gold Coast Shops, Queensland'],
  ['NXT-6UBL-T6JL-J35C-2ZV43', 'Freds Sorting Facility', 'Pacific Hwy, Sydney']
];
```

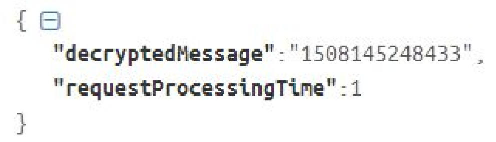
If the QR Code Server verifies that the Producer requesting the QR code is a valid producer, it then decrypts the received encrypted timestamp from the Producer. It does this by contacting the online Nxt node, sending it the data and noce fields received from the Producer. See Task 5 in the above Activity diagram.

Below is a screenshot of what data is being sent to the online Nxt node in order to decrypt the encrypted message from the producer.



```
{
   "requestType":"decryptFrom",
   "secretPhrase":"curve excuse kid content gun horse leap poison girlfriend gaze poison comfort",
   "account":"NXT-QBU9-KSX6-6TH4-H47LR",
   "data":"911921f7a3f76fba0ccfc32f63ddf0bc9c4c2c3f97d8d82747d0b434821af5412591338c37c46ce2f7217806d10ff9f8ca147b4
   "nonce":"ede66cd7dd5acb4a89a6871780cf297091477a0f57b7b58e9b99541dd0de5d07"
}
```

The Nxt node then responds with the text that was encrypted:

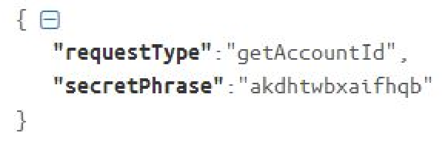


```
{
   "decryptedMessage":"1508145248433",
   "requestProcessingTime":1
}
```

Generation of Nxt Address

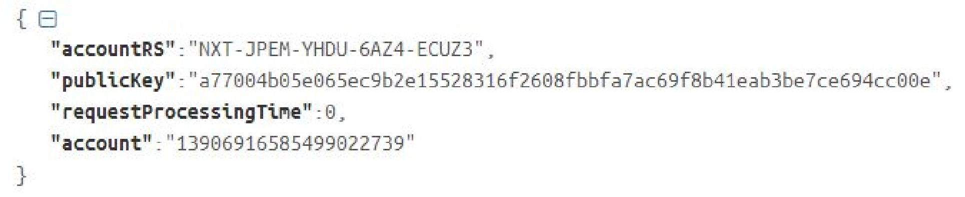
The decrypted message is then checked against previously received QR code requests. If it is a new request, then a new secret phrase is generated, and the Nxt node is contacted again to receive the Nxt address and public key associated with that secret phrase. See Task 5 in the above Activity diagram.

Below is sample data showing what the query to the Nxt node looks like in order to retrieve the account information:



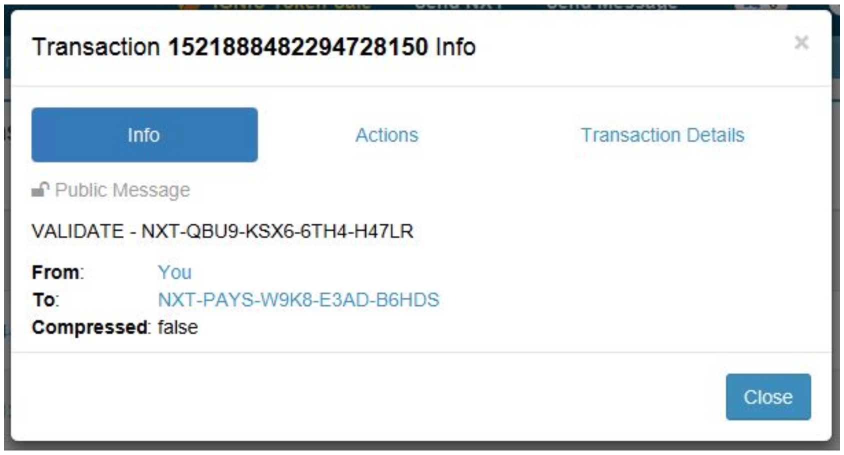
```
{
   "requestType":"getAccountId",
   "secretPhrase":"akdhtwbxaifhqb"
}
```

The response back from the Nxt node contains the account information:



```
{
   "accountRS":"NXT-JPEM-YHDU-6AZ4-ECUZ3",
   "publicKey":"a77004b05e065ec9b2e15528316f2608fbbfa7ac69f8b41eab3be7ce694cc00e",
   "requestProcessingTime":0,
   "account":"13906916585499022739"
}
```

Once the product Nxt address has been received, the ProductChain server sends a `VALIDATE` message to the newly created product to verify that the product has been created by a genuine Producer. The Producer's Nxt address is also placed in the message field to ensure only that producer can first move the product.



Transaction **1521888482294728150** Info

VALIDATE - NXT-QBU9-KSX6-6TH4-H47LR

From: You
To: NXT-PAYS-W9K8-E3AD-B6HDS
Compressed: false

The Nxt account, public key and secret phrase are then converted into a QR Code using the `qr-image` library. This QR Code is in `svg` format.

Caching of Product

Once the QR Code request has been verified it is coming from a valid producer, and the product Nxt address has been generated, the QR code information is sent to the Caching server. See Task 6 in the above Activity diagram. The product Nxt address is added to the list of addresses to be updated from the blockchain, and product information is also stored in a newly created MongoDB collection for each product. This information is publicly available, and can be viewed by anyone wanting to verify product information. See "Publicly Available Data" document.

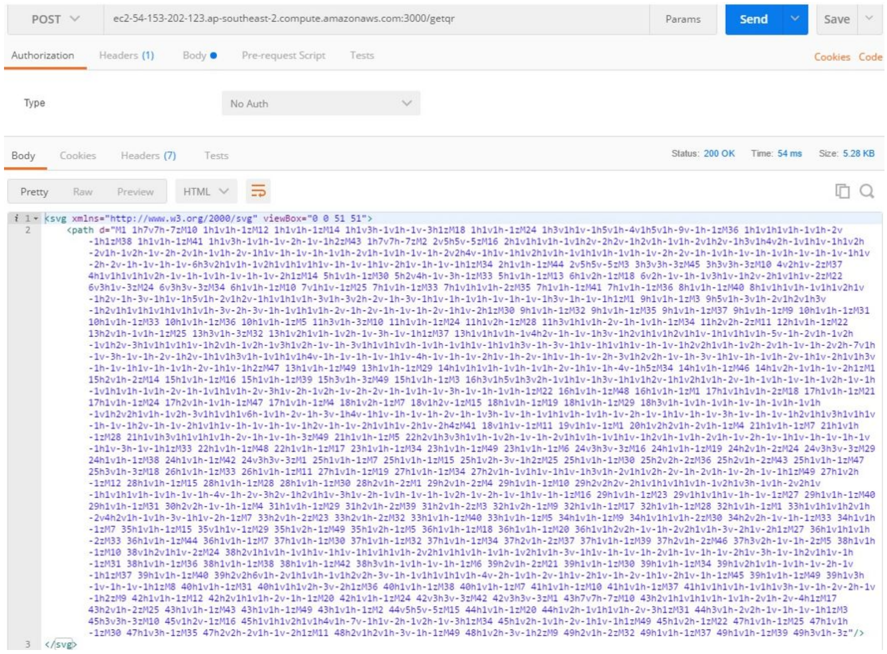
Below is a screenshot of the information the Caching server stored on each product generated.

```
"_id":0,
"qrAddress":"NXT-ZBB9-5YBY-ZGV8-AGMJK",
"qrPubKey":"b43dd92ad1f8f35d33ba558fad9f861426bbd6e98029f360efdc0eb1f14a006c",
"qrPrivKey":"ZoDsiYmVPOl8jii5",
"poducerAddr":"NXT-QBU9-KSX6-6TH4-H47LR",
"productName":"Eggs 6 pack",
"productId":"0000854",
"batchId":"0000986",
"producerName":"John Egg Farm",
"producerLocation":"Croydon Hills, Victoria",
"timestamp":1507104000523
```

Response to Producer

Once the QR Code in the form of an `svg` has been generated, it is sent back to the Producer.

Below is a screenshot of what the response to the Producer looks like.



ec2-54-153-202-123.ap-southeast-2.compute.amazonaws.com:3000/getqr

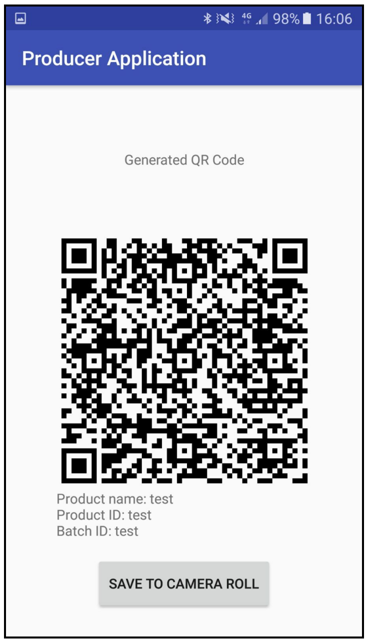
Type: No Auth

Body | Cookies | Headers (7) | Tests

Status: 200 OK   Time: 54 ms   Size: 5.28 KB

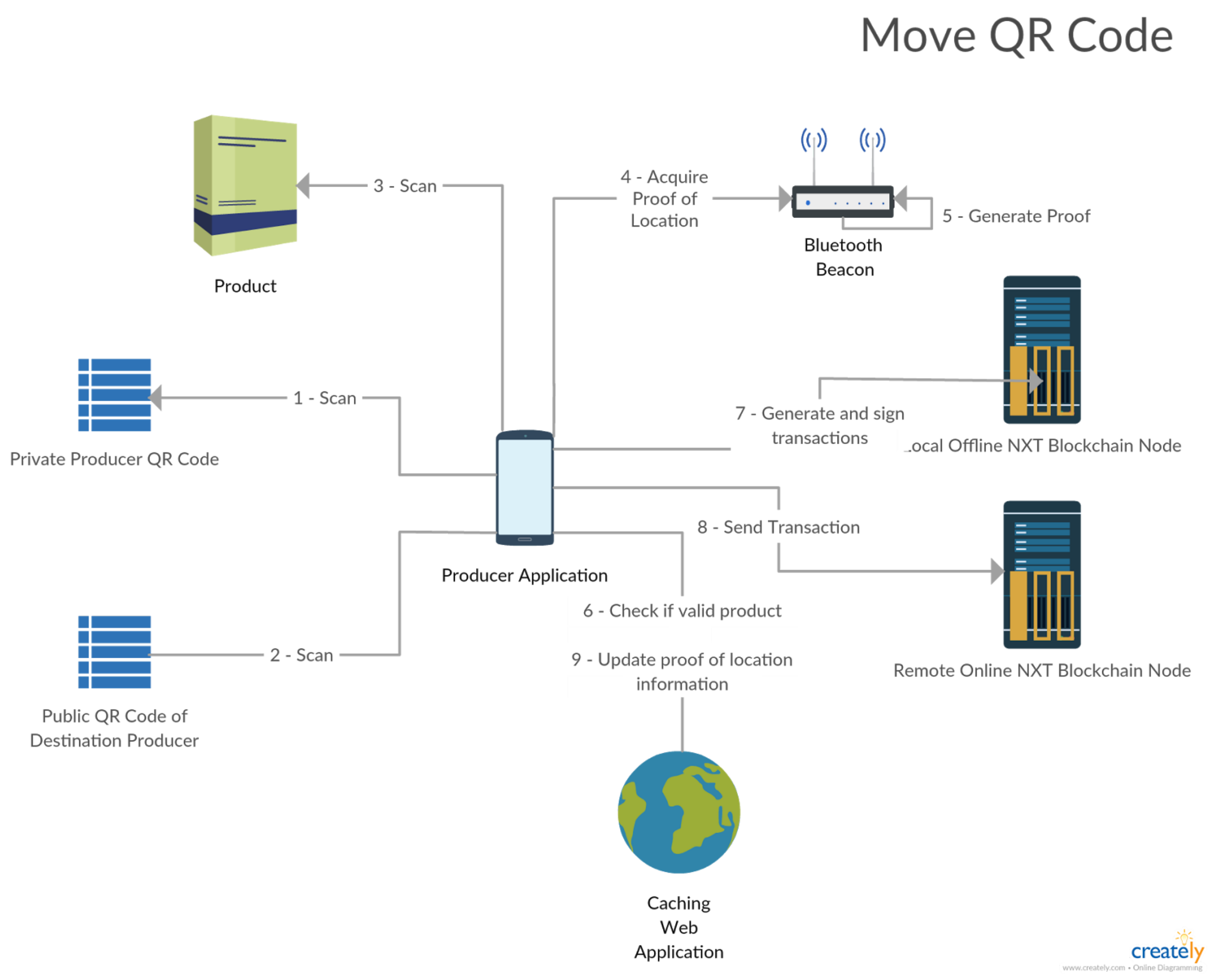
```
<svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 51 51">
    <path d="M1 1h7v7h-7zM10 1h1v1h-1zM12 1h1v1h-1zM14 1h1v3h-1v1h-1v-3h1zM18 1h1v1h-1zM24 1h3v1h1v-1h5v1h-4v1h5v1h-9v-1h-1zM36 1h1v1h1v1h-1v1h-2v
        -1h1zM38 1h1v1h-1zM41 1h1v3h-1v1h-1v-2h-1v-1h2zM43 1h7v7h-7zM2 2v5h5v-5zM16 2h1v1h1v1h-1v1h2v-2h2v-1h2v1h-1v1h-2v1h2v-1h3v1h4v2h-1v1h1v-1h1v2h
        -2v1h-1v2h-1v-2h-2v1h-1v1h-2v-1h1v-1h-1v-1h-1v1h-2v1h-1v1h-1v-1h-2v2h4v-1h1v-1h1v2h1v1h-1v1h1v1h-1v1h-1v1h-1v-2h-2v-1h-1v1h-1v-1h-1v1h-1v-1h-1v-1h1v
        -2h-2v-1h-1v-1h-1v-6h3v2h1v1h-1v2h1v1h1v1h1v-1h-1v-1h1v-2h1v-1h-1v-1h1zM34 2h1v1h-1zM44 2v5h5v-5zM3 3h3v3h-3zM45 3h3v3h-3zM10 4v2h1v-2zM37
        4h1v1h1v1h1v2h-1v-1h-1v1h-1v-1h-1v-2h1zM14 5h1v1h-1zM30 5h2v4h-1v-3h-1zM33 5h1v1h-1zM13 6h1v2h-1zM18 6v2h-1v-1h-1v3h1v-1h2v-2h1v1h1v-2zM22
        6v3h1v-3zM24 6v3h3v-3zM34 6h1v1h-1zM10 7v1h1v-1zM25 7h1v1h-1zM33 7h1v1h1v1h-2zM35 7h1v1h-1zM41 7h1v1h-1zM36 8h1v1h-1zM40 8h1v1h1v1h-1v1h1v2h1v
        -1h2v-1h-3v-1h1v-1h5v1h-2v1h2v-1h1v1h1v1h-3v1h-3v2h-2v-1h-3v-1h1v-1h-1v1h-1v-1h-1v-1h3v-1h-1v-1h1zM1 9h1v1h-1zM3 9h5v1h-3v1h-2v1h2v1h3v
        -1h2v1h1v1h1v1h1v1h1v1h-3v-2h-3v-1h-1v1h1v1h-2v-1h-2v-1h-1v-1h-2v-1h1v-2h1zM30 9h1v1h-1zM32 9h1v1h-1zM35 9h1v1h-1zM37 9h1v1h-1zM9 10h1v1h-1zM31
        10h1v1h-1zM33 10h1v1h-1zM36 10h1v1h-1zM5 11h3v1h-3zM10 11h1v1h-1zM24 11h1v2h-1zM28 11h3v1h1v1h-2v-1h-1v1h-1zM34 11h2v2h-2zM11 12h1v1h-1zM22
        13h2v1h-1v1h-1zM25 13h3v1h-3zM32 13h1v2h1v1h-1v2h-1v-3h-1v-1h1zM37 13h1v1h1v1h-1v4h2v-1h-1v-1h3v-1h2v1h1v1h2v1h1v-1h1v1h1v1h-5v-1h-2v1h-1v2h
        -1v1h2v-3h1v1h1v1h1v-1h2v1h-1v2h-1v3h1v2h-1v-1h-3v1h1v1h1v1h-1v1h-1v1h1v-1h1v1h3v-1h-3v-1h1v-1h1v1h1v-1h-1v-1h2v2h1v1h-1v2h-2v1h-1v-1h-2v2h-7v1h
        -1v-3h-1v-1h-2v-1h2v-1h1v1h3v1h-1v1h1v1h4v-1h-1v-1h-1v-1h1v-4h-1v-1h-1v-2h1v-1h-2v-1h1v-1h-1v-2h-3v1h2v2h-1v-1h-3v-1h1v-1h-1v1h-2v-1h1v-2h1v1h3v
        -1h-1v-1h1v-1h-1v1h-2v-1h1v-1h2zM47 13h1v1h-1zM49 13h1v1h-1zM29 14h1v1h1v1h-1v1h-1v1h-2v-1h1v-1h-4v-1h5zM34 14h1v1h-1zM46 14h1v2h-1v1h-1v-2h1zM1
        15h2v1h-2zM14 15h1v1h-1zM16 15h1v1h-1zM39 15h3v1h-3zM49 15h1v1h-1zM3 16h3v1h5v1h3v2h-1v1h1v-1h3v-1h1v1h2v-1h1v2h1v1h-2v-1h-1v1h-1v-1h-1v2h-1v-1h
        -1v1h1v1h-1v1h-2v-1h-1v1h1v1h-2v-3h1v-2h-1v2h-1v-2h-2v-1h-1v1h-1v-3h-1v-1h-1v1h-1zM22 16h1v1h-1zM48 16h1v1h-1zM1 17h1v1h1v1h-2zM18 17h1v1h-1zM21
        17h1v1h-1zM24 17h2v1h-1v1h-1zM47 17h1v1h-1zM4 18h1v2h-1zM7 18v1h2v-1zM15 18h1v1h-1zM19 18h1v1h-1zM29 18h3v1h-1v1h-1v1h-1v-1h-1v1h-1v1h
        -1v1h2v2h1v1h-1v2h-3v1h1v1h1v6h-1v1h-2v-1h-3v-1h4v-1h1v-1h-1v-1h-2v-1h-1v3h-1v-1h-1v1h1v1h-1v1h-1v-2h-1v-1h1v-1h-1v-3h-1v-1h-1v-1h2v1h1v3h1v1h1v
        -1h-1v-1h2v-1h-1v-2h1v1h1v-1h-1v-1h-1v-1h2v-1h-1v-2h1v1h1v-2h1v-2h4zM41 18v1h1v-1zM11 19v1h1v-1zM1 20h1v2h2v1h-2v1h-1zM4 21h1v1h-1zM7 21h1v1h
        -1zM28 21h1v1h3v1h1v1h1v1h-2v-1h-1v-1h-3zM49 21h1v1h-1zM5 22h2v1h3v3h1v1h-1v2h-1v-1h-2v1h1v1h-1v1h1v-1h2v1h-1v1h-2v1h-1v-2h-1v-1h1v-1h-1v-1h-1v
        -1h1v-3h-1v-1h1zM33 22h1v1h-1zM48 22h1v1h-1zM17 23h1v1h-1zM34 23h1v1h-1zM49 23h1v1h-1zM6 24v3h3v-3zM16 24h1v1h-1zM19 24h2v1h-2zM24 24v3h3v-3zM29
        24h1v1h-1zM38 24h1v1h-1zM42 24v3h3v-3zM1 25h1v1h-1zM7 25h1v1h-1zM15 25h1v2h-3v-1h2zM25 25h1v1h-1zM30 25h2v2h-2zM36 25h2v1h-2zM43 25h1v1h-1zM47
        25h3v1h-3zM18 26h1v1h-1zM33 26h1v1h-1zM11 27h1v1h-1zM19 27h1v1h-1zM34 27h2v1h-1v1h1v-1h1v-1h3v1h-2v1h1v2h-2v-1h-2v1h-1v-2h-1v-1h1zM49 27h1v2h
        -1zM12 28h1v1h-1zM15 28h1v1h-1zM28 28h1v1h-1zM30 28h2v1h-2zM1 29h2v1h-2zM4 29h1v1h-1zM10 29h2v2h2v-2h1v1h1v1h1v1h-1v2h1v3h-1v1h-2v2h1v
        -1h1v1h1v1h-1v1h-1v-1h-4v-1h-2v-3h2v-1h2v1h1v-3h1v-2h-1v1h-1v-1h-1v2h-1v-2h-1v-1h1v-1h-1zM16 29h1v1h-1zM23 29v1h1v1h1v-1h-1v-1zM27 29h1v1h-1zM40
        29h1v1h-1zM31 30h2v2h-1v-1h-1zM4 31h1v1h-1zM29 31h2v1h-2zM39 31h2v1h-2zM3 32h1v2h-1zM9 32h1v1h-1zM17 32h1v1h-1zM28 32h1v1h-1zM1 33h1v1h1v1h2v1h
        -2v4h2v1h-1v1h-3v-1h1v-2h-1zM7 33h2v1h-2zM23 33h2v1h-2zM32 33h1v1h-1zM40 33h1v1h-1zM5 34h1v1h-1zM9 34h1v1h1v1h-2zM30 34h2v2h-1v-1h-1zM33 34h1v1h
        -1zM7 35h1v1h-1zM15 35v1h1v-1zM29 35h1v2h-1zM49 35h1v2h-1zM5 36h1v1h-1zM18 36h1v1h-1zM20 36h1v1h2v2h-1v-1h-2v2h1v1h-3v-2h1v-2h1zM27 36h1v1h1v1h
        -2zM33 36h1v1h-1zM44 36h1v1h-1zM7 37h1v1h-1zM30 37h1v1h-1zM32 37h1v1h-1zM34 37h2v1h-2zM37 37h1v1h-1zM39 37h2v1h-2zM46 37h3v2h-1v-1h-2zM5 38h1v1h
        -1zM10 38v1h2v1h1v-2zM24 38h2v1h1v1h-1v1h1v-1h1v-1h1v1h1v1h-2v2h1v1h1v1h-1v1h-1v2h1v1h-3v-1h1v-1h-1v-1h-2v1h-1v-1h-1v-2h1v-3h-1v-1h2v1h1v-1h
        -1zM31 38h1v1h-1zM36 38h1v1h-1zM38 38h1v1h-1zM42 38h3v1h-1v1h-1v-1h-1zM6 39h2v1h-2zM21 39h1v1h-1zM30 39h1v1h-1zM34 39h1v2h1v1h-1v1h-1v-2h-1v
        -1h1zM37 39h1v1h-1zM40 39h2v2h6v1h-2v1h1v1h-1v1h2v2h-3v-1h-1v1h1v1h1v1h-4v-2h-1v1h-2v-1h1v-2h1v-1h-2v-1h1v-2h1v-1h-1zM45 39h1v1h-1zM49 39h1v3h
        -1v-1h-1v-1h1zM8 40h1v1h-1zM31 40h1v1h1v2h-3v-2h1zM36 40h1v1h-1zM38 40h1v1h-1zM7 41h1v1h-1zM10 41h1v1h-1zM37 41h1v1h1v1h-1v1h1v3h-1v-1h-2v-2h-1v
        -1h2zM9 42h1v1h-1zM12 42h2v1h1v1h-2v-1h-1zM20 42h1v1h-1zM24 42v3h3v-3zM42 42v3h3v-3zM1 43h7v7h-7zM10 43h2v1h1v1h1v1h-1v1h-2v1h-2v-4h1zM17
        43h2v1h-2zM25 43h1v1h-1zM43 43h1v1h-1zM49 43h1v1h-1zM2 44v5h5v-5zM15 44h1v1h-1zM20 44h1v2h-1v1h1v1h-2v-3h1zM31 44h3v1h-2v2h-1v-1h-1v-1h1zM3
        45h3v3h-3zM10 45v1h2v-1zM16 45h1v1h1v2h1v1h4v1h-7v-1h1v-2h-1v2h-1v-3h1zM34 45h1v2h-1v1h-2v-1h1v-1h1zM49 45h1v2h-1zM22 47h1v1h-1zM25 47h1v1h
        -1zM30 47h1v3h-1zM35 47h2v2h-2v1h-1v-2h1zM11 48h2v1h2v1h-3v-1h-1zM49 48h1v2h-3v-1h2zM9 49h2v1h-2zM32 49h1v1h-1zM37 49h1v1h-1zM39 49h3v1h-3z"/>
</svg>
```

The Producer Application renders the `svg` on the screen and allows the user to save it to camera roll as a PNG. This allows the user to then print out the QR code and apply it to a product.



Product name: test
Product ID: test
Batch ID: test

Move QR Code

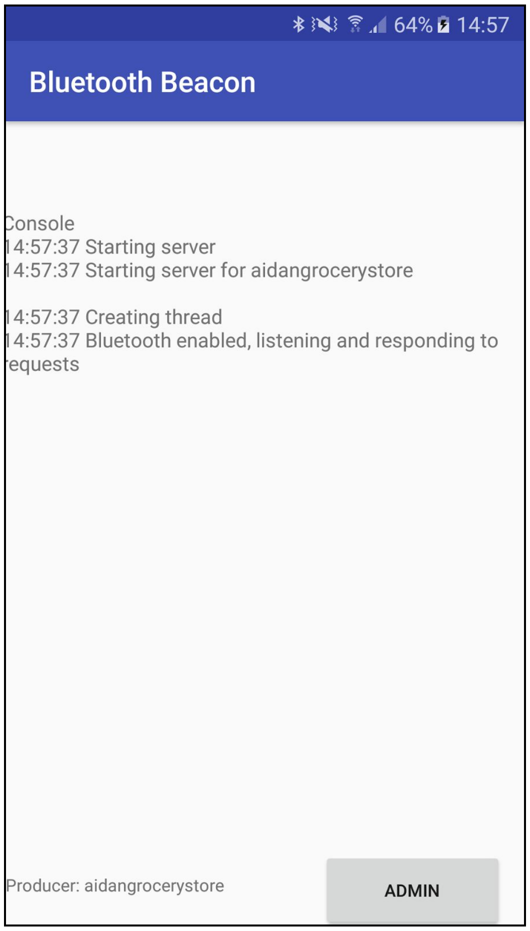

Move QR Code
Product
3 - Scan
4 - Acquire Proof of Location
Bluetooth Beacon
5 - Generate Proof
Private Producer QR Code
1 - Scan
7 - Generate and sign transactions
Local Offline NXT Blockchain Node
8 - Send Transaction
Producer Application
6 - Check if valid product
Public QR Code of Destination Producer
2 - Scan
9 - Update proof of location information
Remote Online NXT Blockchain Node
Caching Web Application
creately

User Scans QR Codes

The user opens the **Producer Application** then presses the *Move Product* button on the home screen. The User will then press *the Scan Producer* (Current Producer private QR), *Scan Product QR* (Product QR) *and Scan Nxt Producer QR* (Next Producer public QR) buttons. Each button will open the ZXing barcode scanner, that will scan the Producer's Private QR code. The Producer application will then retrieve the relevant information from each QR in json.

Bluetooth Beacon

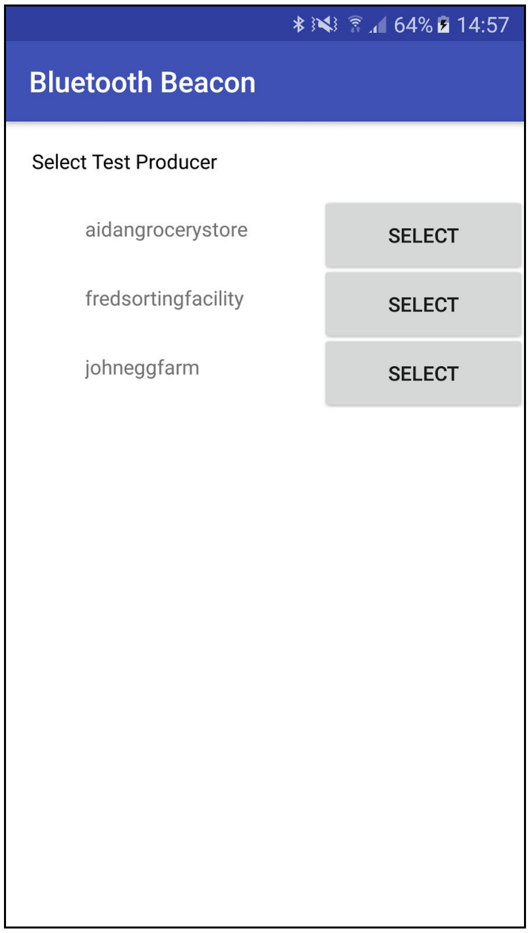
Before Location Proof can be gathered, the bluetooth beacon needs to be started. For testing purposes, it has a *Admin* button, that allows the testing of different Producers on the fly. When the admin button is pressed, the beacon will display all the Producers that have been loaded. Choosing another one will change the name of the producer on the main screen and change the keys being used to validate location.



```
onsole
4:57:37 Starting server
4:57:37 Starting server for aidangrocerystore

4:57:37 Creating thread
4:57:37 Bluetooth enabled, listening and responding to
equests
```

Producer: aidangrocerystore



Select Test Producer

aidangrocerystore	SELECT
fredsortingfacility	SELECT
johneggfarm	SELECT

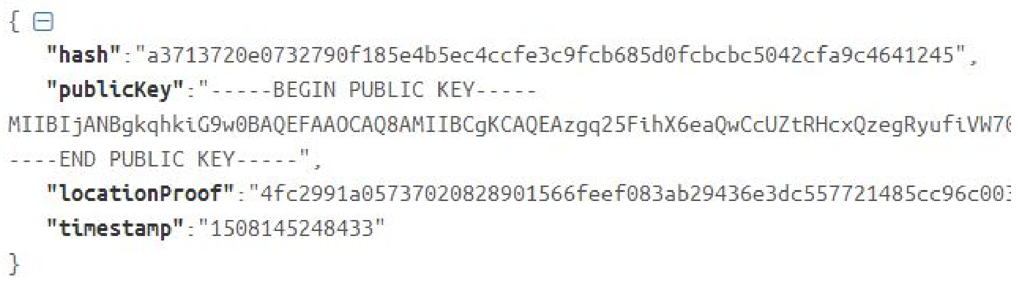
When the application is started a thread running the bluetooth server will start. It sits on the socket.Accept() function waiting for a client to connect. Once a client has connected the beacon executes a number of functions to create the values to send back to the Producer Application.

1. The Beacon gets the current epoch time and creates a digital signature of the timestamp with the Producer's private key. It then verifies it in the same function with the Producer's Public key to ensure the signature is correct.

2. The beacon creates a csv string with the signature, Producer Public Key and timestamp.

3. This csv string is then hashed with sha256 so that the proof size is suitable to store in the NXT blockchain's arbitrary message field.

4. The original csv string containing the signature (locationProof), Producer Public Key (publicKey), timestamp and hash of the data (hash) is sent back to the Producer Application.



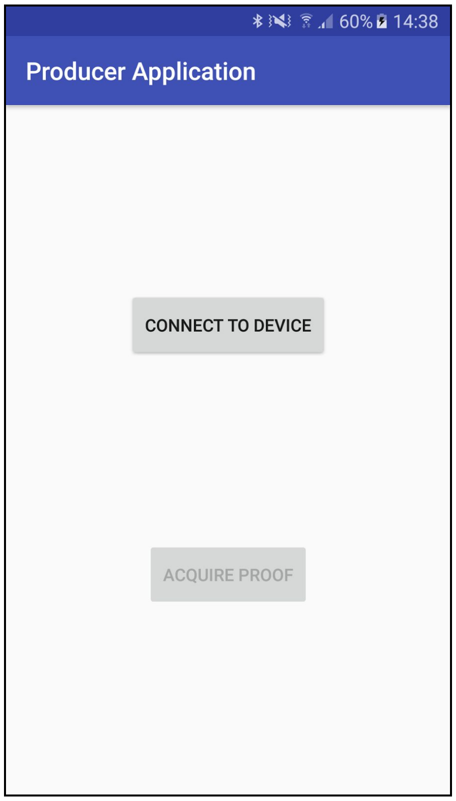
```
{ ⊟
   "hash":"a3713720e0732790f185e4b5ec4ccfe3c9fcb685d0fcbcbc5042cfa9c4641245",
   "publicKey":"-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAzgq25FihX6eaQwCcUZtRHcxQzegRyufiVW7
----END PUBLIC KEY-----",
   "locationProof":"4fc2991a05737020828901566feef083ab29436e3dc557721485cc96c00
   "timestamp":"1508145248433"
}
```

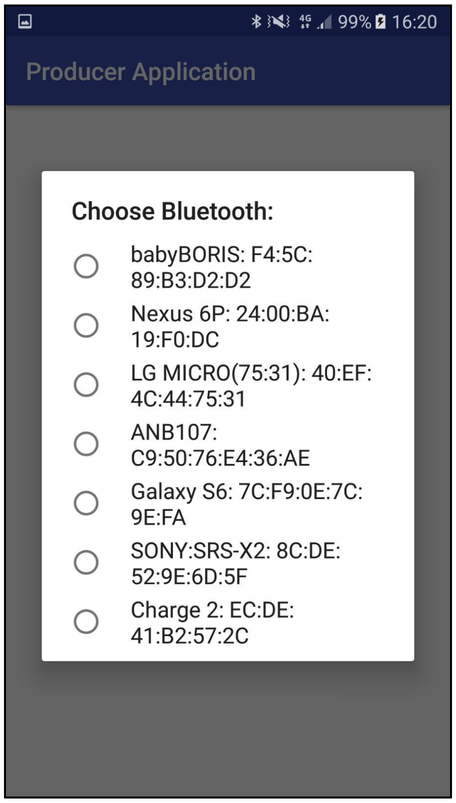
The Beacons socket then closes and reopens another one straight away, ready to accept another connection.

Location Proof

The user will then press *Prove Location* which will open a new Activity. This activity is used for proving location and contacts the bluetooth beacon to get required information to add to the blockchain. The user will press *Connect to Device* which will display a fragment containing all bluetooth devices that have been previously paired to the Android device.

The user will choose which device the Bluetooth Beacon Application is running on from the list..

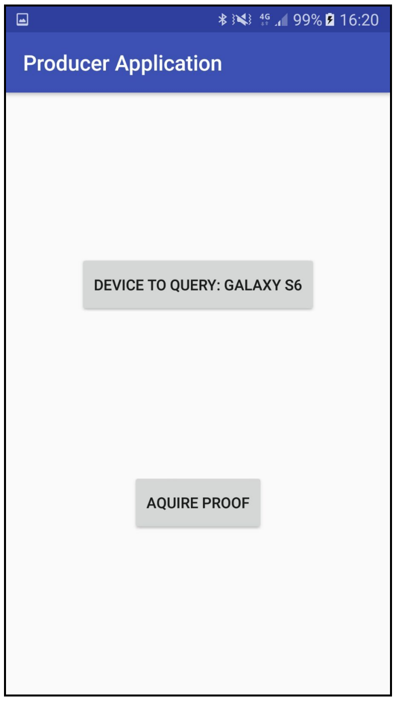


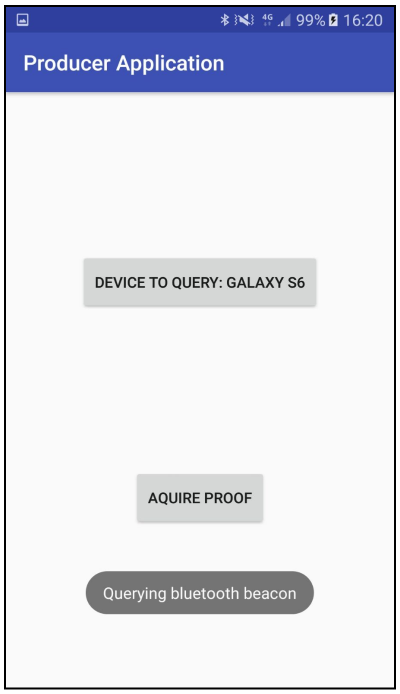


Choose Bluetooth:

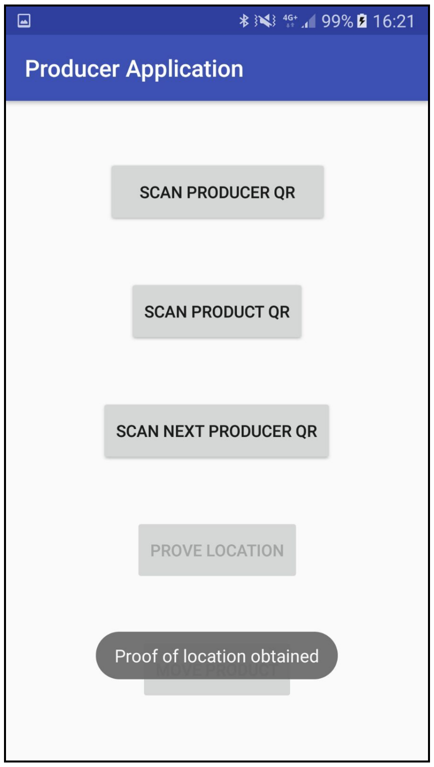
- babyBORIS: F4:5C:89:B3:D2:D2
- Nexus 6P: 24:00:BA:19:F0:DC
- LG MICRO(75:31): 40:EF:4C:44:75:31
- ANB107: C9:50:76:E4:36:AE
- Galaxy S6: 7C:F9:0E:7C:9E:FA
- SONY:SRS-X2: 8C:DE:52:9E:6D:5F
- Charge 2: EC:DE:41:B2:57:2C

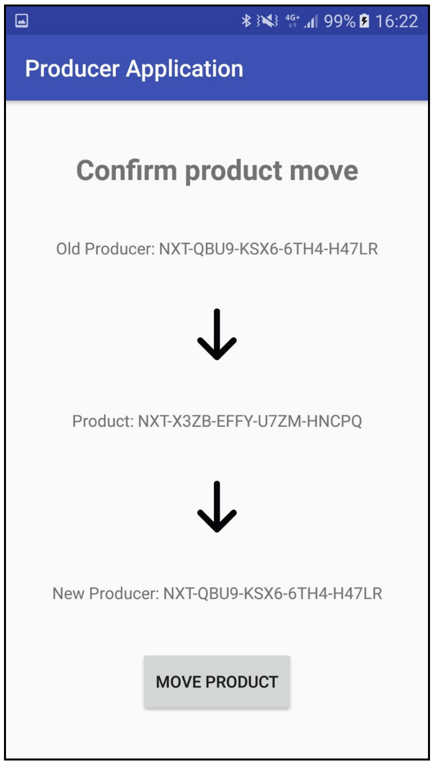
After selecting the device, its name will be displayed as the *Device to Query*. The user will then press the *Acquire Proof* button, which will create a new thread running a bluetooth client and attempt to location and connect to the device running the bluetooth beacon.





If the transmission is successful the beacon will display a Toast. No verification of the location proof is checked on the Producer Application. Once the *Move Product* button is pressed the user is displayed a Confirm move screen, where they can check the correct Producer and Product NXT addresses. When the *Move Product* button is pressed again, numerous transactions will take place.





Confirm product move

Old Producer: NXT-QBU9-KSX6-6TH4-H47LR

Product: NXT-X3ZB-EFFY-U7ZM-HNCPQ

New Producer: NXT-QBU9-KSX6-6TH4-H47LR

Check if Product is Valid

The Producer will send a *HTTP POST* to the Caching Server to check if the created QR code is valid. It sends the Product's NXT account address and the ProductChain NXT Address.

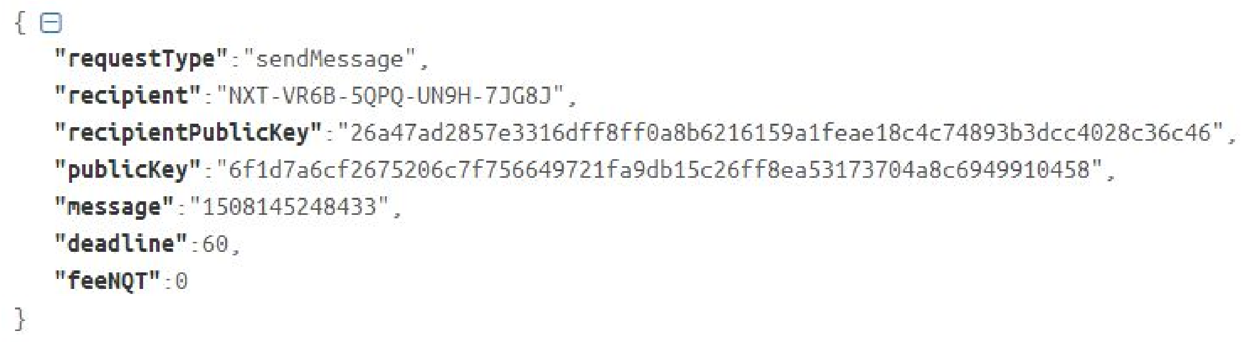
The Caching server then checks to see if the queried product has been validated by the ProductChain Nxt account. See Step 6 in the above Activity diagram. This check is necessary due to the Nxt blockchain generating blocks roughly every minute, and it is not possible for a product to be moved before it has been validated.

The Caching server returns either `true` or `false` depending on whether the Product has been validated yet.

Generate Transaction

If the validation succeeds the Producer will create an offline *MOVE* transaction on the local offline NXT node. This is done by sending the Product NXT account address, Product Public Key, Current Producer Public Key and the Nxt Producer NXT account address.

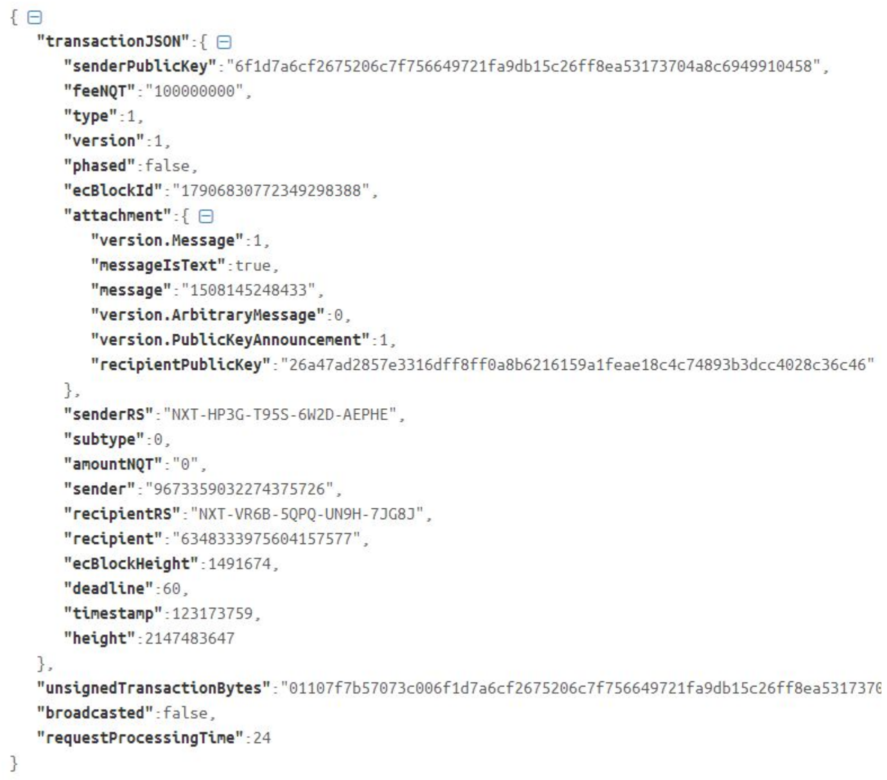
Sent:



```
{
    "requestType":"sendMessage",
    "recipient":"NXT-VR6B-5QPQ-UN9H-7JG8J",
    "recipientPublicKey":"26a47ad2857e3316dff8ff0a8b6216159a1feae18c4c74893b3dcc4028c36c46",
    "publicKey":"6f1d7a6cf2675206c7f756649721fa9db15c26ff8ea53173704a8c6949910458",
    "message":"1508145248433",
    "deadline":60,
    "feeNQT":0
}
```

The Producer application will receive a large message back, we will use the `unsignedTransactionBytes` and sign them offline in the next step.

Response:

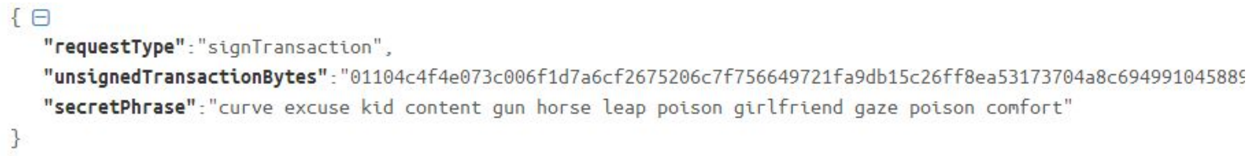


```
{
   "transactionJSON":{
      "senderPublicKey":"6f1d7a6cf2675206c7f756649721fa9db15c26ff8ea53173704a8c6949910458",
      "feeNQT":"100000000",
      "type":1,
      "version":1,
      "phased":false,
      "ecBlockId":"17906830772349298388",
      "attachment":{
         "version.Message":1,
         "messageIsText":true,
         "message":"1508145248433",
         "version.ArbitraryMessage":0,
         "version.PublicKeyAnnouncement":1,
         "recipientPublicKey":"26a47ad2857e3316dff8ff0a8b6216159a1feae18c4c74893b3dcc4028c36c46"
      },
      "senderRS":"NXT-HP3G-T95S-6W2D-AEPHE",
      "subtype":0,
      "amountNQT":"0",
      "sender":"9673359032274375726",
      "recipientRS":"NXT-VR6B-5QPQ-UN9H-7JG8J",
      "recipient":"6348333975604157577",
      "ecBlockHeight":1491674,
      "deadline":60,
      "timestamp":123173759,
      "height":2147483647
   },
   "unsignedTransactionBytes":"01107f7b57073c006f1d7a6cf2675206c7f756649721fa9db15c26ff8ea531737
   "broadcasted":false,
   "requestProcessingTime":24
}
```

Sign Transaction

If the generation of the transaction is successfully, a *HTTP POST* request will be sent to the Local Offline NXT Blockchain node to sign the transaction before it is entered into the live blockchain. The `unsignedTransactionBytes` and the Current Producers Private Key (secretPhrase) are sent in the request.

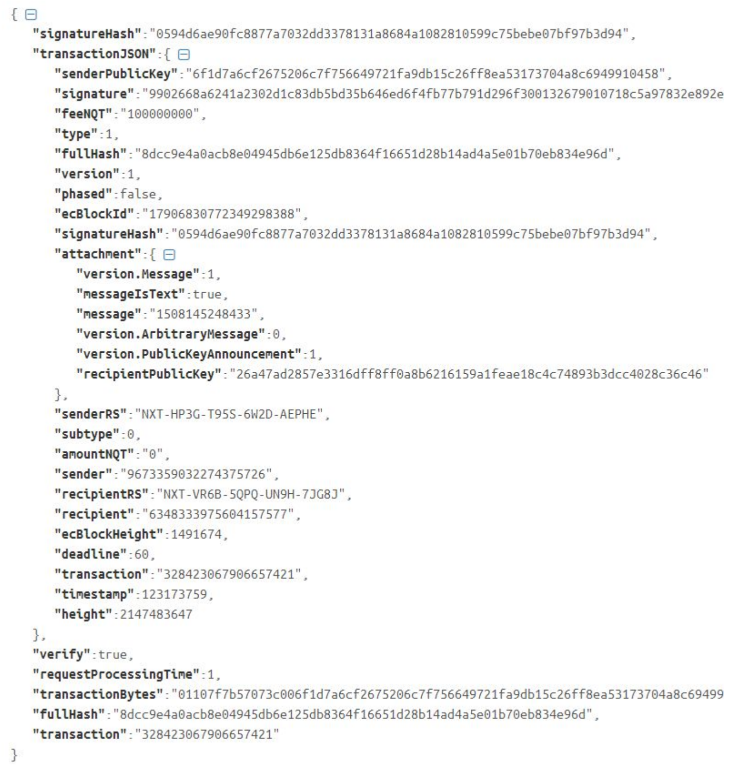
Sent:



```
{
   "requestType":"signTransaction",
   "unsignedTransactionBytes":"01104c4f4e073c006f1d7a6cf2675206c7f756649721fa9db15c26ff8ea53173704a8c69499104588
   "secretPhrase":"curve excuse kid content gun horse leap poison girlfriend gaze poison comfort"
}
```

The Producer Application will receive the signed transaction bytes (`transactionBytes`) back, read to be added to the live blockchain.

Response:

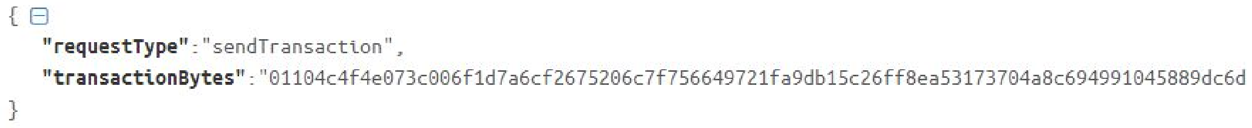


```
{
   "signatureHash":"0594d6ae90fc8877a7032dd3378131a8684a1082810599c75bebe07bf97b3d94",
   "transactionJSON":{
      "senderPublicKey":"6f1d7a6cf2675206c7f756649721fa9db15c26ff8ea53173704a8c6949910458",
      "signature":"9902668a6241a2302d1c83db5bd35b646ed6f4fb77b791d296f300132679010718c5a97832e892e
      "feeNQT":"100000000",
      "type":1,
      "fullHash":"8dcc9e4a0acb8e04945db6e125db8364f16651d28b14ad4a5e01b70eb834e96d",
      "version":1,
      "phased":false,
      "ecBlockId":"17906830772349298388",
      "signatureHash":"0594d6ae90fc8877a7032dd3378131a8684a1082810599c75bebe07bf97b3d94",
      "attachment":{
         "version.Message":1,
         "messageIsText":true,
         "message":"1508145248433",
         "version.ArbitraryMessage":0,
         "version.PublicKeyAnnouncement":1,
         "recipientPublicKey":"26a47ad2857e3316dff8ff0a8b6216159a1feae18c4c74893b3dcc4028c36c46"
      },
      "senderRS":"NXT-HP3G-T95S-6W2D-AEPHE",
      "subtype":0,
      "amountNQT":"0",
      "sender":"9673359032274375726",
      "recipientRS":"NXT-VR6B-5QPQ-UN9H-7JG8J",
      "recipient":"6348333975604157577",
      "ecBlockHeight":1491674,
      "deadline":60,
      "transaction":"328423067906657421",
      "timestamp":123173759,
      "height":2147483647
   },
   "verify":true,
   "requestProcessingTime":1,
   "transactionBytes":"01107f7b57073c006f1d7a6cf2675206c7f756649721fa9db15c26ff8ea53173704a8c69499
   "fullHash":"8dcc9e4a0acb8e04945db6e125db8364f16651d28b14ad4a5e01b70eb834e96d",
   "transaction":"328423067906657421"
}
```

Send Transaction

Once the transaction has been generated and signed, it is then sent to the remote online NXT blockchain node to be distributed onto the blockchain.

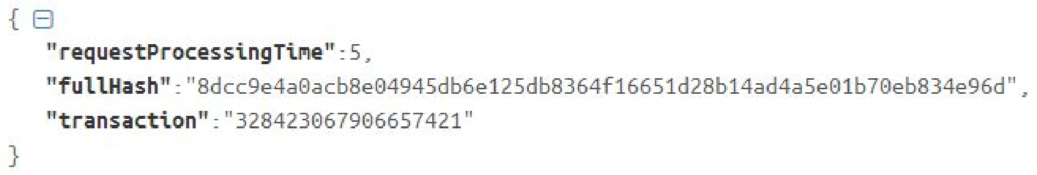
Sent:



```
{
   "requestType":"sendTransaction",
   "transactionBytes":"01104c4f4e073c006f1d7a6cf2675206c7f756649721fa9db15c26ff8ea53173704a8c694991045889dc6d
}
```

The response is received and indicates that the transaction was added to the blockchain. If an error code appears the transaction was not added correctly. The Producer Application checks if the `fullHash` exists then goes on to update the Proof of Location.

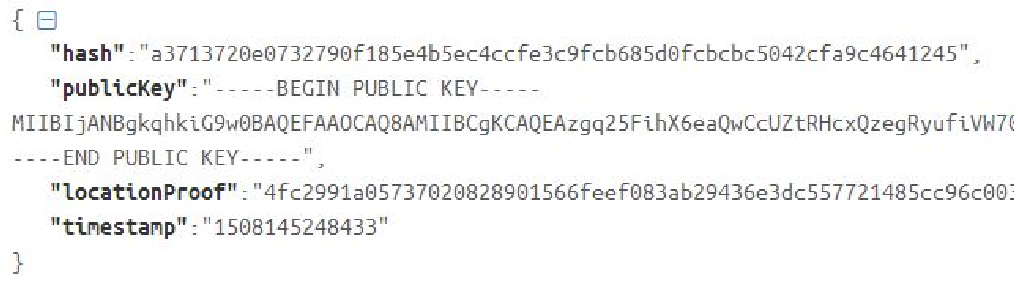
Response:



```
{
   "requestProcessingTime":5,
   "fullHash":"8dcc9e4a0acb8e04945db6e125db8364f16651d28b14ad4a5e01b70eb834e96d",
   "transaction":"328423067906657421"
}
```

Update Proof of Location Information

The final step in moving a Product is to update the proof of location information in the Caching Server. A *HTTP POST* is sent to the Caching Server with the proof of location hash (`hash`), public key (`publicKey`), signature (`locationProof`) and timestamp.

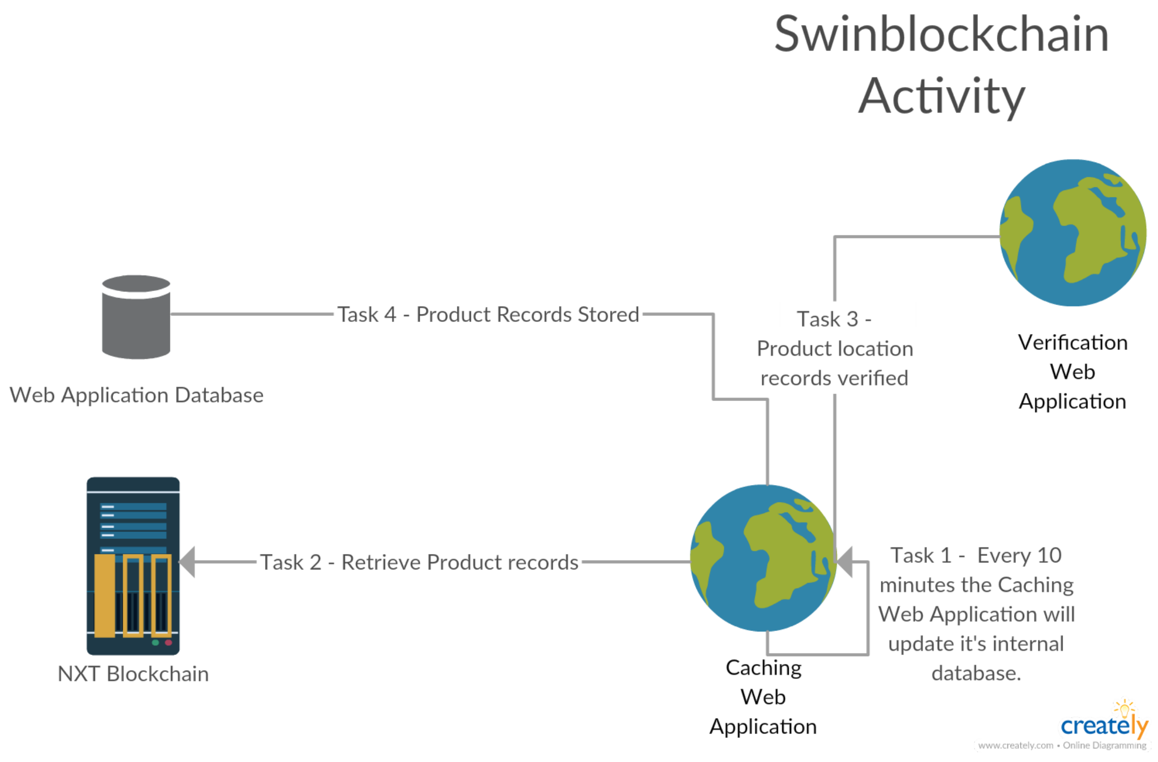


```
{
   "hash":"a3713720e0732790f185e4b5ec4ccfe3c9fcb685d0fcbcbc5042cfa9c4641245",
   "publicKey":"-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAzgq25FihX6eaQwCcUZtRHcxQzegRyufiVW7
----END PUBLIC KEY-----",
   "locationProof":"4fc2991a05737020828901566feef083ab29436e3dc557721485cc96c00
   "timestamp":"1508145248433"
}
```

Once the Caching server receives this information, it updates the hashInfo Collection on MongoDB. This Collection contains the publicly available information from all of the `MOVE` requests, and is contacted when verifying the hash of the location information which is stored in the blockchain.It then returns `true` to the Producer Application.

If the response equals `true`, the Producer Application will display a Toast to the user and return them to the home screen.

Caching Server Updates & Verification


Swinblockchain Activity
Task 4 - Product Records Stored
Web Application Database
Task 3 - Product location records verified
Verification Web Application
Task 2 - Retrieve Product records
NXT Blockchain
Task 1 - Every 10 minutes the Caching Web Application will update it's internal database.
Caching Web Application
creately
www.creately.com • Online Diagramming

The Caching server stores information about the products, as well as the location proof information sent to it by the Producer application when products are moved. It regularly contacts the Nxt blockchain, and receives transactions to each of the products stored. This information can be considered publicly available information (see “Publicly Available Data” document), and was implemented in this way for easily accessibility for both the web servers and the mobile applications.

Regular Updates

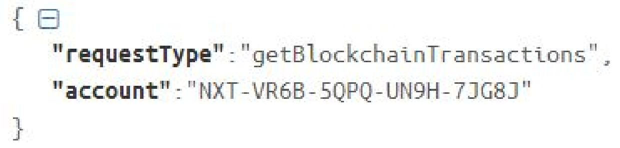
The Caching Server regularly contacts the Nxt blockchain to retrieve updated information about product movements and newly validated products. See Task 1 in the above Activity diagram.

As the Nxt blockchain generates blocks roughly every minute, newly created products may not be fully updated the first time they are queried. The caching server in our implementation is updated every 30 seconds, but can easily be changed (see “Set up Caching Server” document).

Product Retrieval, Query, & Validation

The caching server goes through its list of cached products, and queries the Nxt blockchain for each product. This query receives an array of transactions that were made to the product. See Task 2 in the above Activity diagram.

Below is a screenshot of the POST request made to the blockchain node which returns the array of transactions.

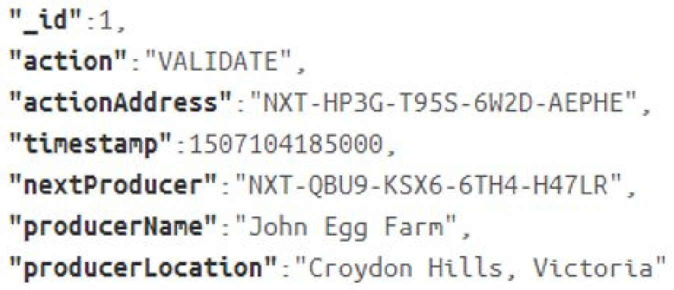


```
{ ⊟
   "requestType":"getBlockchainTransactions",
   "account":"NXT-VR6B-5QPQ-UN9H-7JG8J"
}
```

Each transaction is then checked to see if it is a `VALIDATE` transaction (from the ProductChain address) or if it is a `MOVE` transaction (from a valid Producer address).

If the transaction is a `VALIDATE` transaction, then the Caching server queries the QR Code Server to retrieve the associated producer name for the address listed in the VALIDATE message. This information is then added to the MongoDB Collection for this specific product.

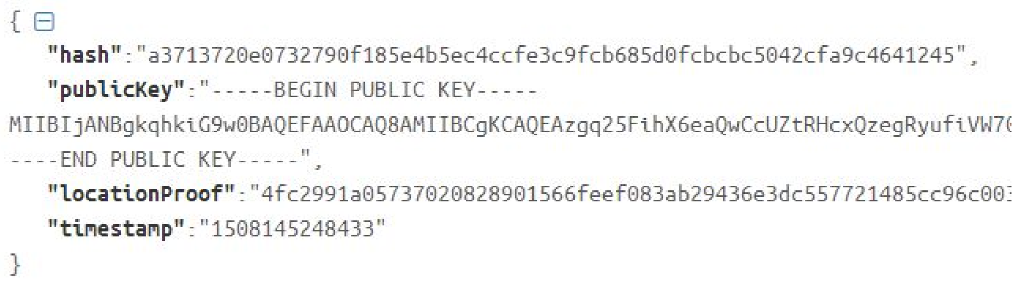
Below you can see what this `VALIDATE` transaction looks like when it has been added to the Product's Collection:



```
"_id":1,
"action":"VALIDATE",
"actionAddress":"NXT-HP3G-T95S-6W2D-AEPHE",
"timestamp":1507104185000,
"nextProducer":"NXT-QBU9-KSX6-6TH4-H47LR",
"producerName":"John Egg Farm",
"producerLocation":"Croydon Hills, Victoria"
```

If it is a MOVE transaction, then the Caching server must contact the Verification server in order to verify if the hash of the location-proof information in the transaction is valid. It does this by first querying the `hashInfo` Collection of publicly available information about the location proofs (see document "Publicly Available Data"). Once it has received the RSA hash, signature and timestamp from the `hashInfo` Collection, it sends this to the Verification server to verify if the information is correct. See Task 3 in the above Activity diagram.

Below is a screenshot of the information sent to the Verification Server:



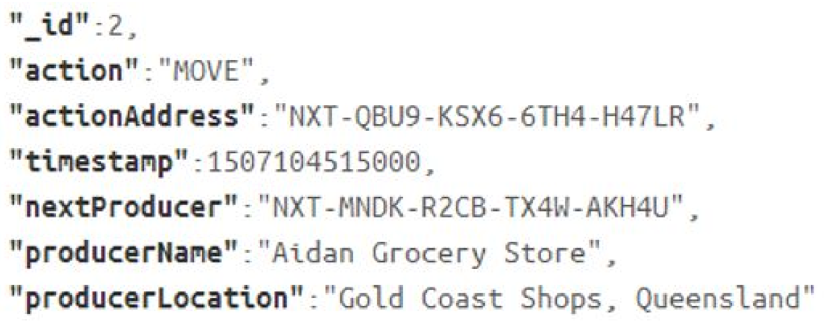
```
{ ⊟
   "hash":"a3713720e0732790f185e4b5ec4ccfe3c9fcb685d0fcbcbc5042cfa9c4641245",
   "publicKey":"-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAzgq25FihX6eaQwCcUZtRHcxQzegRyufiVW7
----END PUBLIC KEY-----",
   "locationProof":"4fc2991a05737020828901566feef083ab29436e3dc557721485cc96c00
   "timestamp":"1508145248433"
}
```

When this information is received by the Verification server, the location proof, RSA public key and the timestamp are hashed using the sha256 hashing algorithm. The hash of this should match the hash that was generated by the Bluetooth beacon. If it does, then the program continues on to verify the location proof.

The Verification server holds the publicly available information about the Bluetooth beacons. That is the RSA public keys and location of each beacon. In order to verify that the location proof data was signed by the RSA private key, the `node-rsa` library is used. Only if this verification returns successful does the location information get verified. The Verification server then returns `true` or `false` if the location information was verified successfully or unsuccessfully.

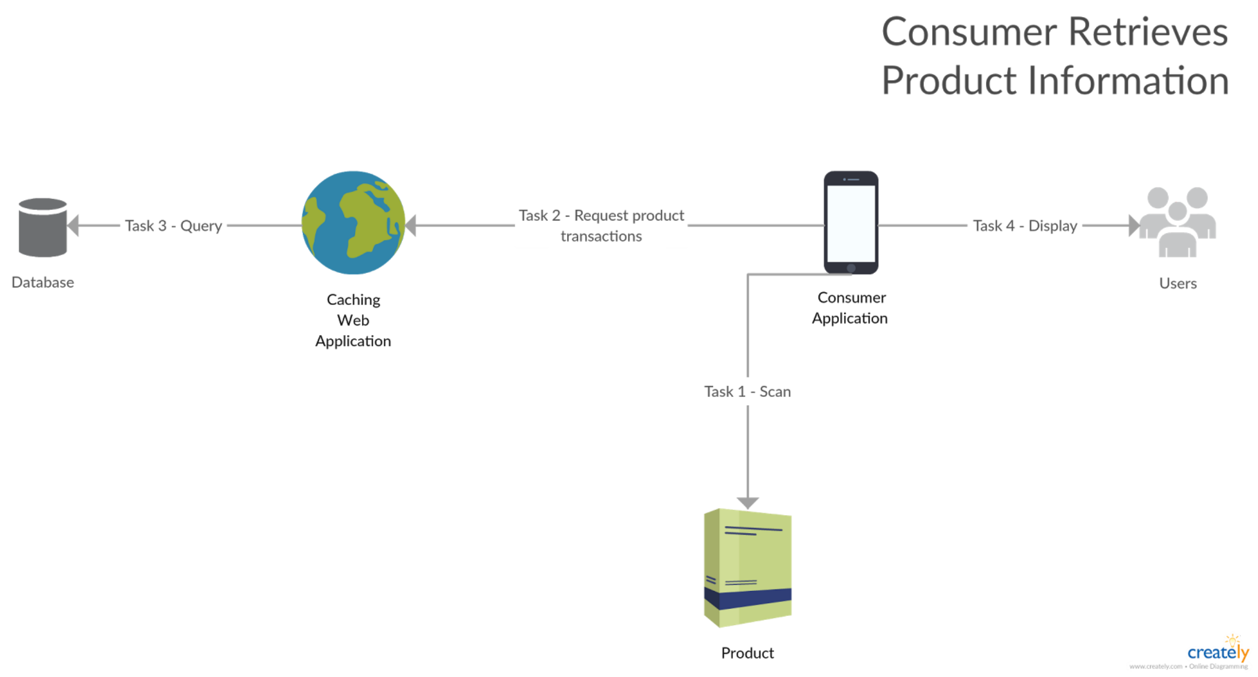
Finally, if the `MOVE` transaction was verified successfully, the transaction is added to the product Collection.

Below is a screenshot of what this information looks like. See Task 4 in the above Activity diagram.



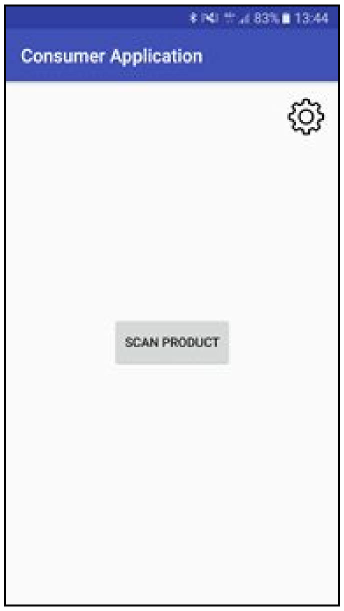
```
"_id":2,
"action":"MOVE",
"actionAddress":"NXT-QBU9-KSX6-6TH4-H47LR",
"timestamp":1507104515000,
"nextProducer":"NXT-MNDK-R2CB-TX4W-AKH4U",
"producerName":"Aidan Grocery Store",
"producerLocation":"Gold Coast Shops, Queensland"
```

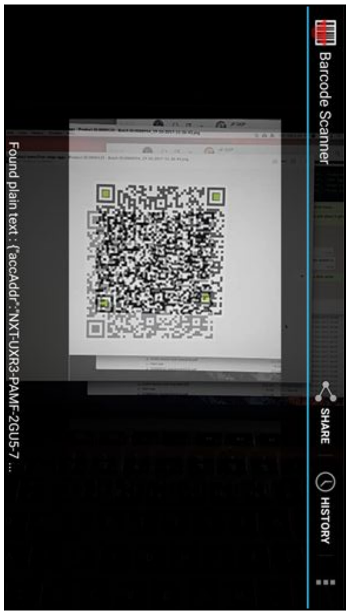
Consumer Retrieves Product Information



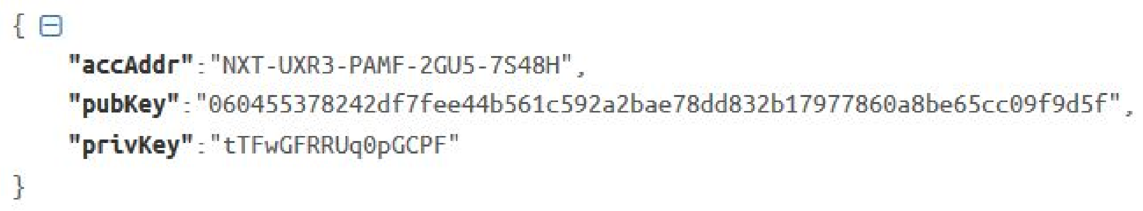
Scan QR Code

The user opens the **Consumer Application** then presses the *Scan Product* button on the home screen. This will open the ZXing barcode scanner, that will scan the Product QR code. The Consumer application will then retrieve the Product's NXT account, Private key and Public key from the QR code in json format.





Found plain text : {"accAddr":"NXT-UXR3-PAMF-2GU5-7 ...



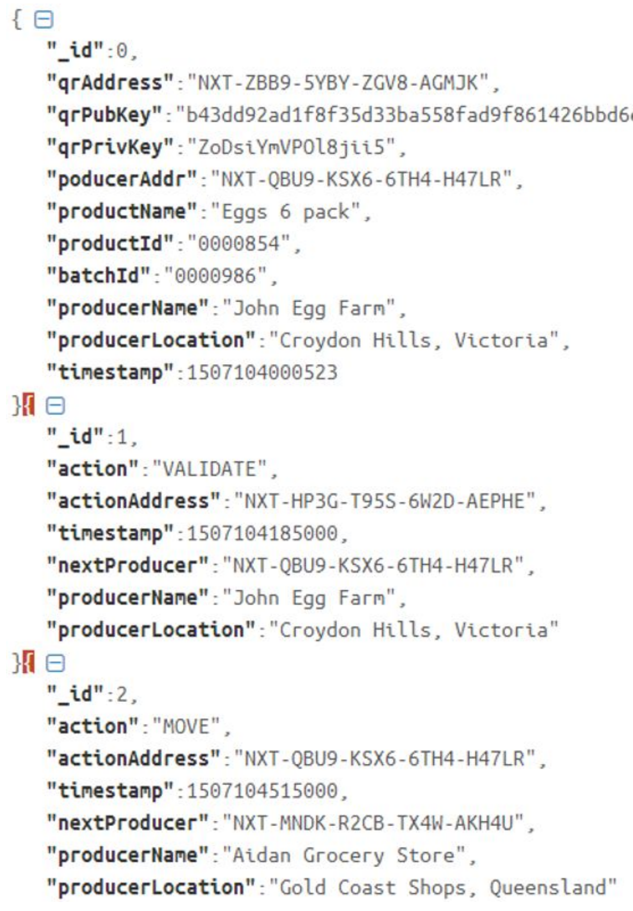
```
{
    "accAddr":"NXT-UXR3-PAMF-2GU5-7S48H",
    "pubKey":"060455378242df7fee44b561c592a2bae78dd832b17977860a8be65cc09f9d5f",
    "privKey":"tTFwGFRRUq0pGCPF"
}
```

Request Product Transactions

The **Consumer Application** will make an *HTTP GET* to the **Caching Server** with the Product's NXT address.

When the product's address is received, the Caching server finds the MongoDB Collection for that specific product. This Collection contains all of the cached information about the product such as the product name, product ID, batch ID, and all of the `VALIDATE` and `MOVE` transactions made to it. It then returns all this information back to the consumer.

Below is a screenshot showing the information returned to a consumer requesting product information.



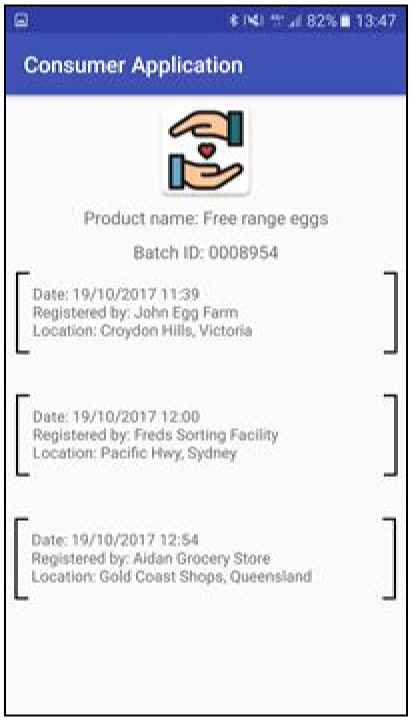
```
{
   "_id":0,
   "qrAddress":"NXT-ZBB9-5YBY-ZGV8-AGMJK",
   "qrPubKey":"b43dd92ad1f8f35d33ba558fad9f861426bbd6e98029f360efdc0eb1f14a006c",
   "qrPrivKey":"ZoDsiYmVPOl8jii5",
   "poducerAddr":"NXT-QBU9-KSX6-6TH4-H47LR",
   "productName":"Eggs 6 pack",
   "productId":"0000854",
   "batchId":"0000986",
   "producerName":"John Egg Farm",
   "producerLocation":"Croydon Hills, Victoria",
   "timestamp":1507104000523
}{
   "_id":1,
   "action":"VALIDATE",
   "actionAddress":"NXT-HP3G-T95S-6W2D-AEPHE",
   "timestamp":1507104185000,
   "nextProducer":"NXT-QBU9-KSX6-6TH4-H47LR",
   "producerName":"John Egg Farm",
   "producerLocation":"Croydon Hills, Victoria"
}{
   "_id":2,
   "action":"MOVE",
   "actionAddress":"NXT-QBU9-KSX6-6TH4-H47LR",
   "timestamp":1507104515000,
   "nextProducer":"NXT-MNDK-R2CB-TX4W-AKH4U",
   "producerName":"Aidan Grocery Store",
   "producerLocation":"Gold Coast Shops, Queensland"
```

Request Producer Transactions

The **Consumer Application** will then receive the product information and details about each producer that has made a transaction to that product. The data is checked to ensure it is valid json and each received Producer is added to an ArrayList.

Display Information

This data is then displayed to the user in a separate, scrollable activity where the user will be able to see information about the product and the time and date each Producer made a transaction to it.



Product name: Free range eggs

Batch ID: 0008954

Date: 19/10/2017 11:39
Registered by: John Egg Farm
Location: Croydon Hills, Victoria

Date: 19/10/2017 12:00
Registered by: Freds Sorting Facility
Location: Pacific Hwy, Sydney

Date: 19/10/2017 12:54
Registered by: Aidan Grocery Store
Location: Gold Coast Shops, Queensland

Improvements

- Implementation of a user management system for creation and distribution of keys for new users. This would also reduce the need to update keys manually on the Bluetooth Beacon, Verification and QR Code Web applications. It would also allow for faster scanning when creating a new Product or moving a Product with the Producer Application.
- The publicly available information such as Producer Nxt address, location information, generated product information etc. are currently being stored on the Caching/QR Code/Verification servers. This information represents information that may be distributed by the Producers and ProductChain. This information could instead be distributed across the producers and received by other means.
- Currently, The Bluetooth Beacon is running on a Android device and can be easily moved. It will need to be installed securely and tamper-proof. The beacon device also may require updated keys at times, which may prove difficult to update. Further research will need to be done into this area.
- The system can be improved by making the Producers update products upon arrival at their location. You would therefore know both when a product left a producer and when it arrived.

Limitations

- Bluetooth beacons require the user of the Producer application to be within Bluetooth distance of the beacon. For Producers who operate over a large area, multiple Bluetooth beacons may be required to cover the whole location. The current system only accounts for one beacon at each site.
- Currently if a valid location proof for a producer is sent by a different producer from the one who was supposed to sent it, it can possibly go through. This needs looking into.