SyncForce x Blockchain

Proof of Concept

Summary, Analysis and Advises

Kasper Hämäläinen / 3203328

11.01.2019

Table of Content

Background

This document is part of Open Innovation- course and project “SyncForce x Blockchain” at Fontys University of Applied Sciences. Main goal of the project is to do research and give advices how SyncForce could benefit from Blockchain technology.

Introduction

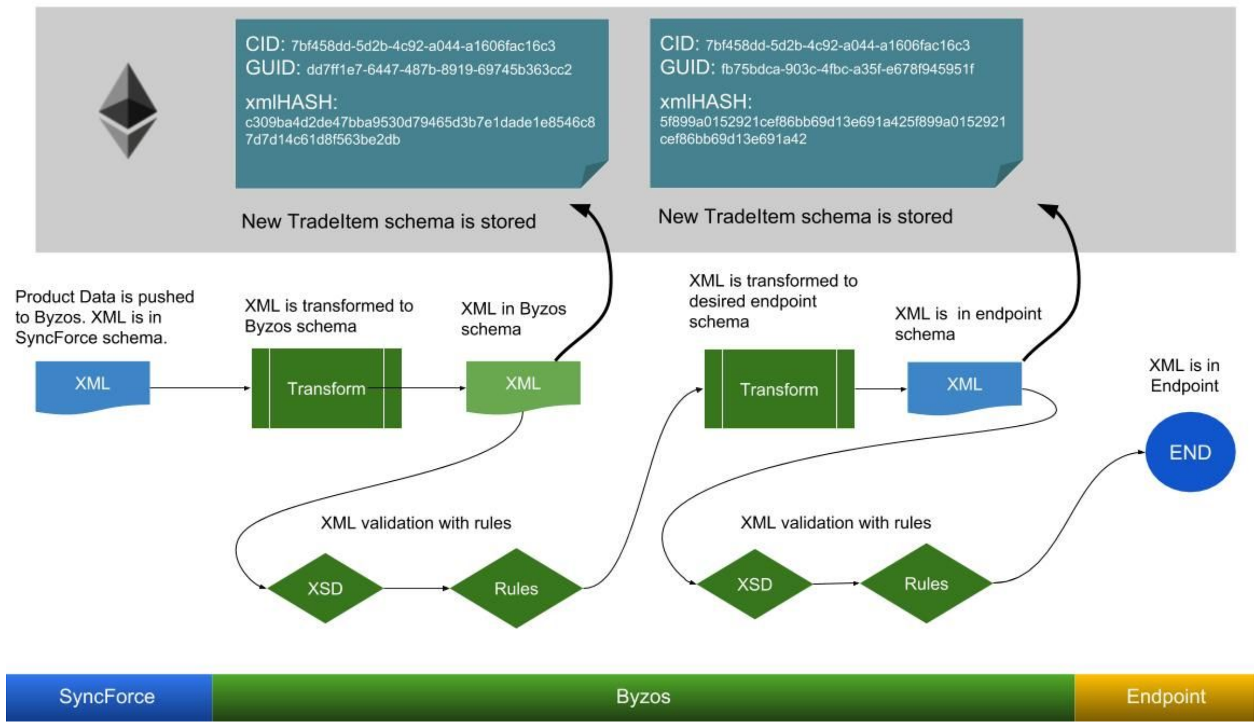
Goal of this document is to provide summary of the functionality within created PoC and what benefits it could give for current processes in SyncForce. Document also states possible future benefits and advices what could be achieved by extending functionality of the PoC and by using Blockchain in general.

Proof Of Concept

Concept

Concept for the PoC is based on identified challenge within Byzos:
"*Comparing, retrieving and managing different versions and schemas of released product information is complicated with Byzos, because versioning is distributed to different schemas.*"

Core idea of the created PoC is that Byzos continue working as it is currently functioning; validating, transforming and syndicating xml-schemas to the wanted endpoints. Blockchain is implemented in a way, that in wanted step, schema is hashed and it is stored to the Blockchain via Smart Contract. From Blockchain, this hash of the product data can be retrieved and from it can be verified authenticity of the product data, as it was in time of the storing.

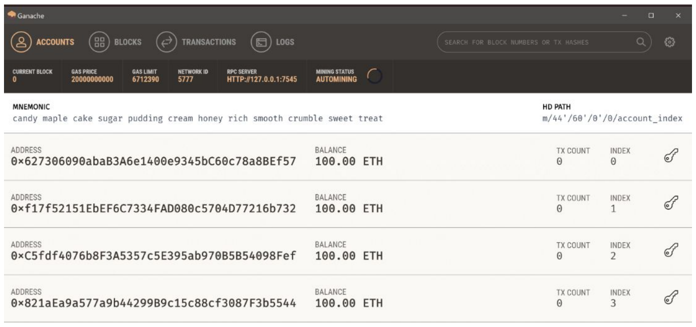

CID: 7bf458dd-5d2b-4c92-a044-a1606fac16c3
GUID: dd7ff1e7-6447-487b-8919-69745b363cc2
xmlHASH: c309ba4d2de47bba9530d79465d3b7e1dade1e8546c87d7d14c61d8f563be2db
New TradeItem schema is stored
CID: 7bf458dd-5d2b-4c92-a044-a1606fac16c3
GUID: fb75bdca-903c-4fbc-a35f-e678f945951f
xmlHASH: 5f899a0152921cef86bb69d13e691a425f899a0152921cef86bb69d13e691a42
New TradeItem schema is stored
Product Data is pushed to Byzos. XML is in SyncForce schema.
XML
XML is transformed to Byzos schema
Transform
XML in Byzos schema
XML
XML validation with rules
XSD
Rules
XML is transformed to desired endpoint schema
Transform
XML is in endpoint schema
XML
XML validation with rules
XSD
Rules
XML is in Endpoint
END
SyncForce
Byzos
Endpoint

Components and used technologies

Ethereum

As identified from the outcome of Byzos Concept document, Ethereum was chosen to use in this project.

To setup local Ethereum network, Ganache software from Truffle Suite is used.



```
CURRENT BLOCK  GAS PRICE    GAS LIMIT  NETWORK ID  RPC SERVER              MINING STATUS
0              20000000000  6712390    5777        HTTP://127.0.0.1:7545   AUTOMINING

MNEMONIC                                                                     HD PATH
candy maple cake sugar pudding cream honey rich smooth crumble sweet treat   m/44'/60'/0'/0/account_index

ADDRESS                                     BALANCE      TX COUNT  INDEX
0×627306090abaB3A6e1400e9345bC60c78a8BEf57  100.00 ETH   0         0

ADDRESS                                     BALANCE      TX COUNT  INDEX
0×f17f52151EbEF6C7334FAD080c5704D77216b732  100.00 ETH   0         1

ADDRESS                                     BALANCE      TX COUNT  INDEX
0×C5fdf4076b8F3A5357c5E395ab970B5B54098Fef  100.00 ETH   0         2

ADDRESS                                     BALANCE      TX COUNT  INDEX
0×821aEa9a577a9b44299B9c15c88cf3087F3b5544  100.00 ETH   0         3
```

Mini-Byzos

To mimic functionality of the Byzos, software was developed by using C# and Nethereum- library. Usage of these technologies was requirement from the project stakeholder, because Byzos is written in C#. Nethereum is C# compatible library, which establish communication with Ethereum network.

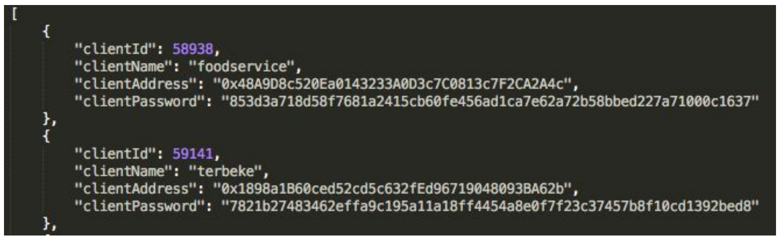
Functionality

Client Registry

Each client has public address which correspond addresses in Ethereum network, created in Ganache. Each address has corresponding password, respecting public-private key cryptography used in Ethereum.

Client address is used to link ownership of Trade Items to the clients and password is used to sign the transaction and when schema is hashed. This approach adds an extra layer of security to the process.

For future, this client registry should be stored in external service, like Azure Key Vault.

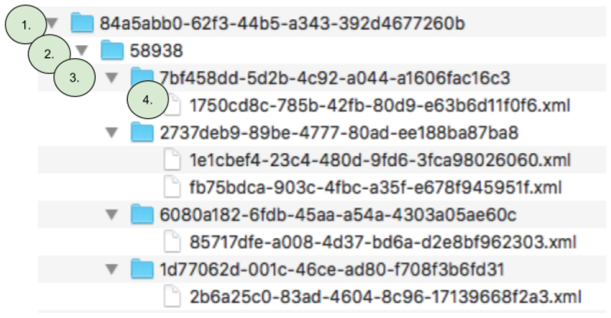


```
[
    {
        "clientId": 58938,
        "clientName": "foodservice",
        "clientAddress": "0x48A9D8c520Ea0143233A0D3c7C0813c7F2CA2A4c",
        "clientPassword": "853d3a718d58f7681a2415cb60fe456ad1ca7e62a72b58bbed227a71000c1637"
    },
    {
        "clientId": 59141,
        "clientName": "terbeke",
        "clientAddress": "0x1898a1B60ced52cd5c632fEd96719048093BA62b",
        "clientPassword": "7821b27483462effa9c195a11a18ff4454a8e0f7f23c37457b8f10cd1392bed8"
    },
```

Trade Item Registry

To mimic original Byzos functionality xml-schemas are already “transformed” and stored in local drive. Software has a function, which reads folder structure and creates needed object, which can then be stored to Ethereum.

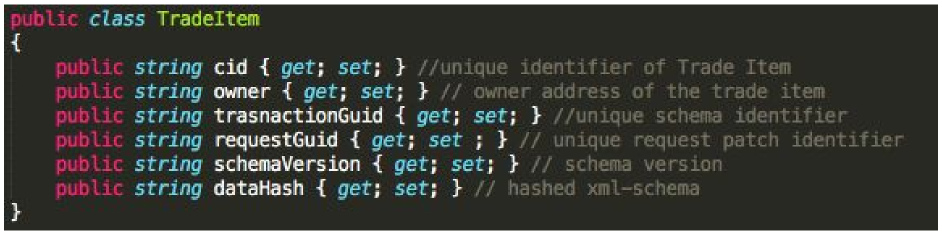
Data from folder structure:
1. requestGuid
2. clientID
3. cid
4. transactionGuid



```
84a5abb0-62f3-44b5-a343-392d4677260b
  58938
    7bf458dd-5d2b-4c92-a044-a1606fac16c3
      1750cd8c-785b-42fb-80d9-e63b6d11f0f6.xml
    2737deb9-89be-4777-80ad-ee188ba87ba8
      1e1cbef4-23c4-480d-9fd6-3fca98026060.xml
      fb75bdca-903c-4fbc-a35f-e678f945951f.xml
    6080a182-6fdb-45aa-a54a-4303a05ae60c
      85717dfe-a008-4d37-bd6a-d2e8bf962303.xml
    1d77062d-001c-46ce-ad80-f708f3b6fd31
      2b6a25c0-83ad-4604-8c96-17139668f2a3.xml
```

Data from xml- file:
- schemaVersion
- dataHash (XML- file is hashed by using HMAC SHA256)

Object:



```
public class TradeItem
{
    public string cid { get; set; } //unique identifier of Trade Item
    public string owner { get; set; } // owner address of the trade item
    public string trasnactionGuid { get; set; } //unique schema identifier
    public string requestGuid { get; set ; } // unique request patch identifier
    public string schemaVersion { get; set; } // schema version
    public string dataHash { get; set; } // hashed xml-schema
}
```

Storing to the Blockchain

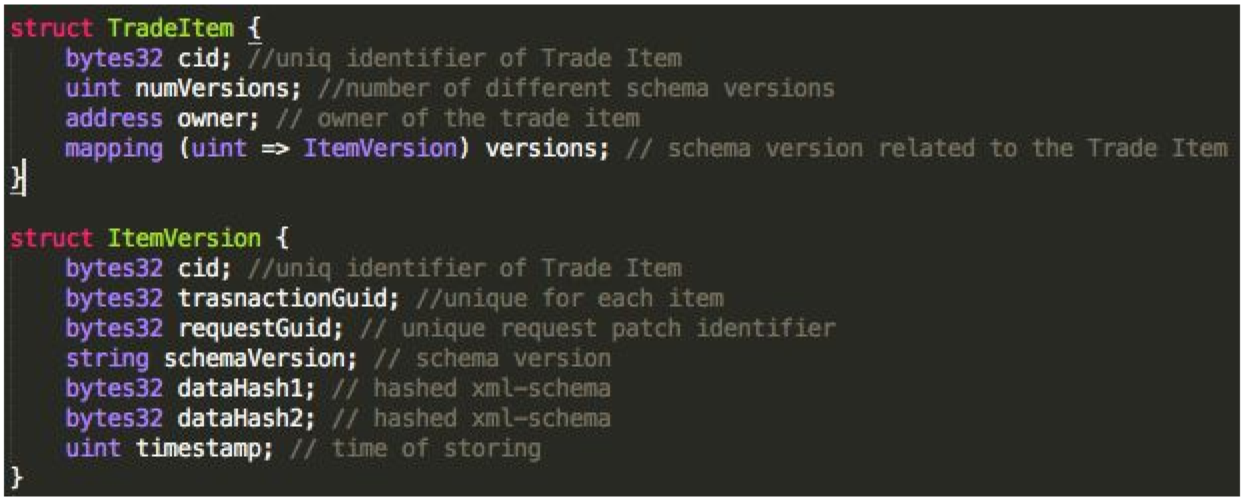
Using Nethereum- library object is stored to Ethereum via Smart Contract. This happens by calling smart contract address and functions in it. Smart Contracts are deployed to the Ethereum network by using Truffle- framework from Truffle Suite.

Ethereum Smart Contracts are written in Solidity, which is Ethereum’s own programming language. In the smart contract is needed functions and models to store Trade Item and related schemas.

Because in Ethereum eventually all data is in bytes level writing and storing data in bytes is recommended. With this approach, unnecessary, slow and costly encoding and decoding can be avoided. During the storing (making transaction) needed data conversions are done in software by Nethereum- library where these data converting functionalities are build-in.

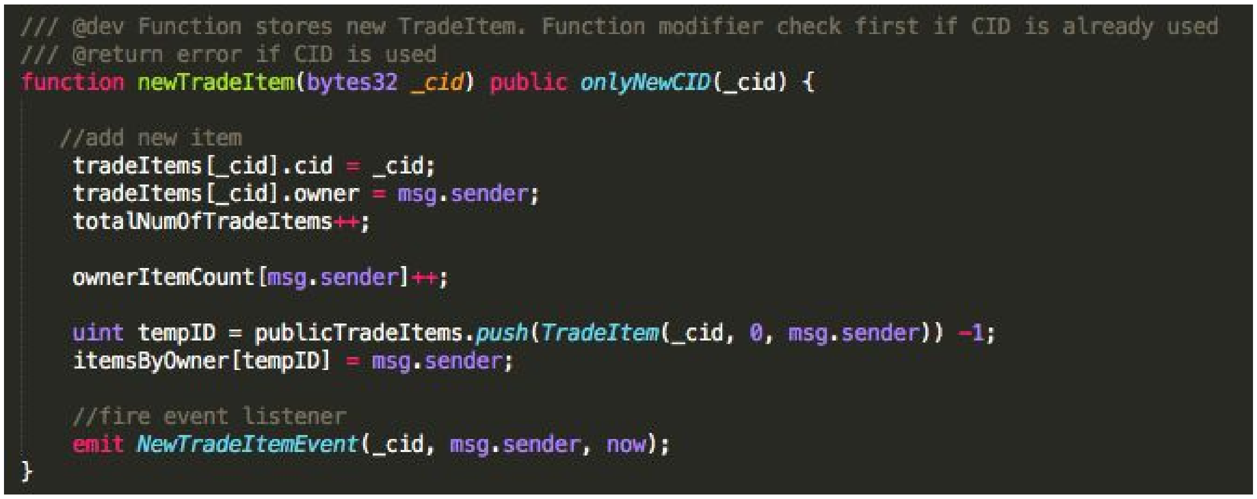
It is notable, that because of the current limitation of Ethereum, bytes32 is largest supported length in bytes variables and because of this schema hash is splitted in two parts. Using string-data types in Solidity is not recommended, but in sake of clarity for this PoC, string was used.

Models in Smart Contract:



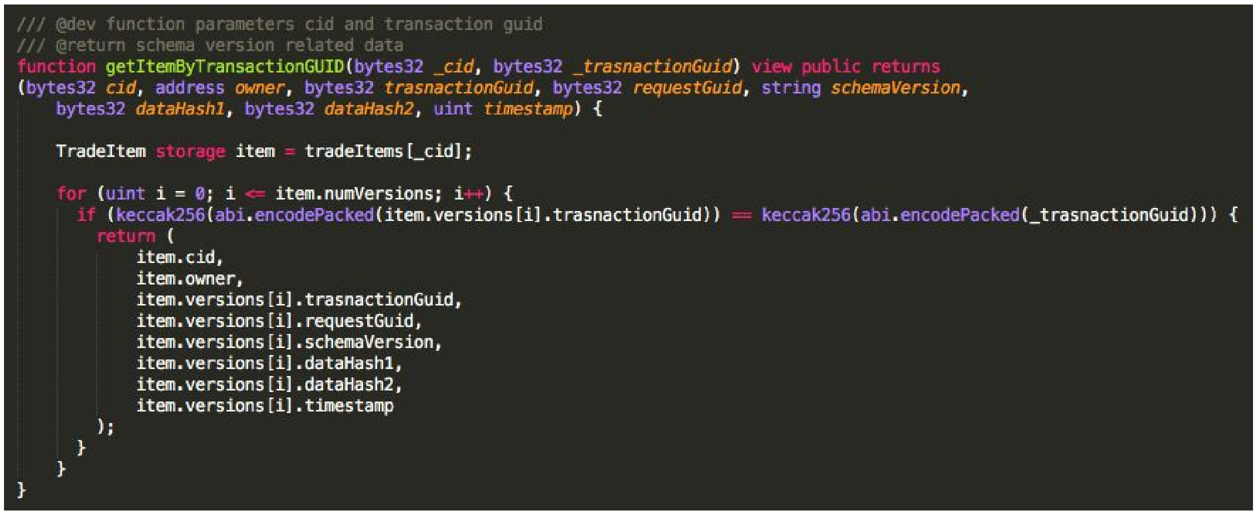
```
struct TradeItem {
    bytes32 cid; //uniq identifier of Trade Item
    uint numVersions; //number of different schema versions
    address owner; // owner of the trade item
    mapping (uint => ItemVersion) versions; // schema version related to the Trade Item
}

struct ItemVersion {
    bytes32 cid; //uniq identifier of Trade Item
    bytes32 trasnactionGuid; //unique for each item
    bytes32 requestGuid; // unique request patch identifier
    string schemaVersion; // schema version
    bytes32 dataHash1; // hashed xml-schema
    bytes32 dataHash2; // hashed xml-schema
    uint timestamp; // time of storing
}
```

New Trade Item function in Smart Contract:



```
/// @dev Function stores new TradeItem. Function modifier check first if CID is already used
/// @return error if CID is used
function newTradeItem(bytes32 _cid) public onlyNewCID(_cid) {

   //add new item
    tradeItems[_cid].cid = _cid;
    tradeItems[_cid].owner = msg.sender;
    totalNumOfTradeItems++;

    ownerItemCount[msg.sender]++;

    uint tempID = publicTradeItems.push(TradeItem(_cid, 0, msg.sender)) -1;
    itemsByOwner[tempID] = msg.sender;

    //fire event listener
    emit NewTradeItemEvent(_cid, msg.sender, now);
}
```

Get schema function in Smart Contract:



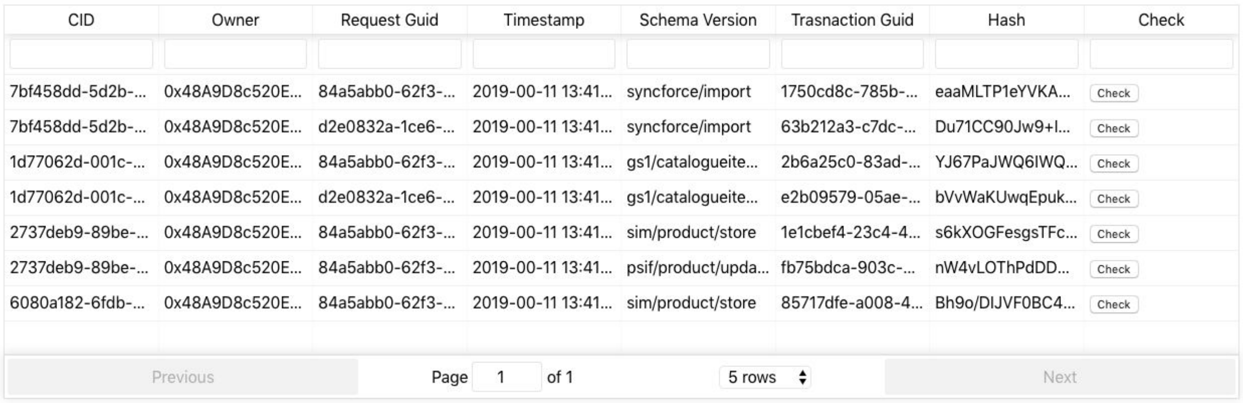
```
/// @dev function parameters cid and transaction guid
/// @return schema version related data
function getItemByTransactionGUID(bytes32 _cid, bytes32 _trasnactionGuid) view public returns
(bytes32 cid, address owner, bytes32 trasnactionGuid, bytes32 requestGuid, string schemaVersion,
    bytes32 dataHash1, bytes32 dataHash2, uint timestamp) {

    TradeItem storage item = tradeItems[_cid];

    for (uint i = 0; i <= item.numVersions; i++) {
      if (keccak256(abi.encodePacked(item.versions[i].trasnactionGuid)) == keccak256(abi.encodePacked(_trasnactionGuid))) {
        return (
            item.cid,
            item.owner,
            item.versions[i].trasnactionGuid,
            item.versions[i].requestGuid,
            item.versions[i].schemaVersion,
            item.versions[i].dataHash1,
            item.versions[i].dataHash2,
            item.versions[i].timestamp
        );
      }
    }
}
```

More details of this transaction functionality can be seen from the project files itself.

Reading data from Blockchain

To retrieve and compare hash of stored Trade Items light web service was created by using ReactJS. To establish connection with Ethereum network Web3.js library is used. This library is provided by Ethereum.

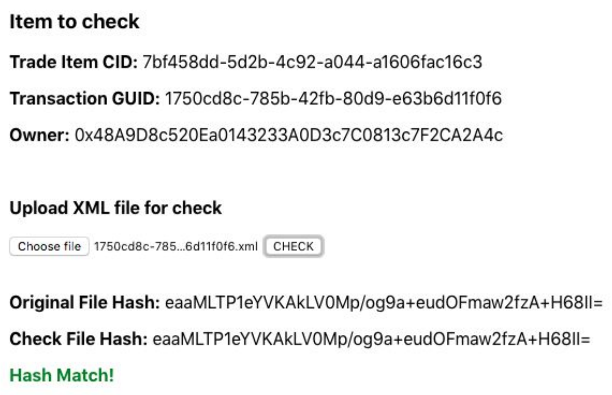
Web service has table, from which all stored Trade Item schemas and information of the it can be seen.



CID	Owner	Request Guid	Timestamp	Schema Version	Trasnaction Guid	Hash	Check
7bf458dd-5d2b-...	0x48A9D8c520E...	84a5abb0-62f3-...	2019-00-11 13:41...	syncforce/import	1750cd8c-785b-...	eaaMLTP1eYVKA...	Check
7bf458dd-5d2b-...	0x48A9D8c520E...	d2e0832a-1ce6-...	2019-00-11 13:41...	syncforce/import	63b212a3-c7dc-...	Du71CC90Jw9+I...	Check
1d77062d-001c-...	0x48A9D8c520E...	84a5abb0-62f3-...	2019-00-11 13:41...	gs1/catalogueite...	2b6a25c0-83ad-...	YJ67PaJWQ6IWQ...	Check
1d77062d-001c-...	0x48A9D8c520E...	d2e0832a-1ce6-...	2019-00-11 13:41...	gs1/catalogueite...	e2b09579-05ae-...	bVvWaKUwqEpuk...	Check
2737deb9-89be-...	0x48A9D8c520E...	84a5abb0-62f3-...	2019-00-11 13:41...	sim/product/store	1e1cbef4-23c4-4...	s6kXOGFesgsTFc...	Check
2737deb9-89be-...	0x48A9D8c520E...	84a5abb0-62f3-...	2019-00-11 13:41...	psif/product/upda...	fb75bdca-903c-...	nW4vLOThPdDD...	Check
6080a182-6fdb-...	0x48A9D8c520E...	84a5abb0-62f3-...	2019-00-11 13:41...	sim/product/store	85717dfe-a008-4...	Bh9o/DIJVF0BC4...	Check

Previous Page 1 of 1 5 rows Next

Web service allows user to upload xml-schema at the browser and then compare hash of it against stored hash in Blockchain.


Item to check

Trade Item CID: 7bf458dd-5d2b-4c92-a044-a1606fac16c3

Transaction GUID: 1750cd8c-785b-42fb-80d9-e63b6d11f0f6

Owner: 0x48A9D8c520Ea0143233A0D3c7C0813c7F2CA2A4c

Upload XML file for check

Original File Hash: eaaMLTP1eYVKAkLV0Mp/og9a+eudOFmaw2fzA+H68Il=

Check File Hash: eaaMLTP1eYVKAkLV0Mp/og9a+eudOFmaw2fzA+H68Il=

Hash Match!

Benefits and challenges

Benefits

With created PoC and within functionality it offers SyncForce can;

- Verify and proof that published and stored schema in Azure servers matches schema which was originally published in that time. This improves data transparency and authenticity.
- Byzos can gross check hash in Blockchain with the possible new schema version. If hash matches, there is no need to store new schema version or there is mistake in process. This endorse process functionality and minimize data redundancy in Azure.
- SyncForce can reproduce once executed request route and related schema transformations to confirm functionality of Byzos. This improves data quality and accuracy.
- Published Trade Items are linked to the each customer, which improves ownership authenticity and can be used to verify intellectual property.
- Stored items in Blockchain can be seen as a “absolute” truth of published trade items. Using this information can be created new models and ways to manage published data.
- Blockchain is immutable audit log of published schemas, it increases transparency and trust in the process.

Identified benefits are notable additions to the current process and can increase customer trust and can improve Byzos service promise.

Needed addition

During the project phase of designing concept main idea was to encode, encrypt and store schema versions as a whole to Ethereum. This plan hit the wall after testing and more thorough research. In Ethereum is maximum Gas which can be used to execute transaction; current xml-schemas, as they are, or encoded are too big to store.

So, limitation with the created PoC is that Blockchain doesn't actual hold the actual product data, only parts of it, from which can be identified which Trade Item it is. Addition would be also store endpoint location of the product data itself, which currently is stored in Azure. This could have been easily implemented to the PoC, but was seen as an unnecessary addition. With this addition, web and other services could use Blockchain as main source of product data and to get necasserty info of where to pull "rest of the data". This itself, is already a good extension and advice would be to explore new models of data management and distribution around it.

To extend by storing more, actual product data as an individual variables in to Blockchain in un advisable, taken into account nature of the Blockchain, which is that it is not meant to be data storage for large files or large data structures. It is advisable that SyncForce carefully think what data is needed to benefit from Blockchain and that stored data stream doesn't increase to unmanageable size.

Using Public Ethereum network

It is notable, that benefits listed earlier are identified as in concept level and more research has to be done, so that this concept and it benefits can be scaled to the production. Open question is solution behind production version of Ethereum network; How costly is to run private Ethereum network and will usage of it advocate benefits of Blockchain? Could there be permissioned consortium network? Will benefits of using public Ethereum network overcome overhead of expenses related to the transactions?

Usage of public Ethereum network can be seen as a one solution for the production. Downside of this is that each transaction cost Gas, which is paid in ETH, cryptocurrency used in Ethereum network. To get overview, what would costs to use public Ethereum network following calculations were made (based on ETH Gas station calculator 11.01.2019):

Action	Gas used	Price in ETH	Price in €
Smart Contract creation (one time or update)	1 816 539	0.0054496	0.69755
Store one new Trade Item CID or Schema	220 699	0.0005484	0.0691

Based on those transaction fees:

Amount of stored schemas:	Cost €
10 000	691,00 €
50 000	3 455,00 €
1 000 000	69 100,00 €

Currently, in one month thru Byzos is processed around 20 000 requests and inside those are possible multiple schemas (one estimate was around 100 000/month). Addition to this is possible import requests.

So, right now; no brainer, it is not advisable to use public Ethereum to store **all** schemas. Notable, is that in case of Ethereum, they are moving to Proof of Stake consensus protocol, which will decrease amount of transaction fees in public network. Also, by tweaking the smart contract struct, cost of transaction can be lowered (usage of strings is costly, minimize data needed to store etc.).

Also, it is notable, that these transaction costs can be avoided when using private or permissioned network. Then the question will be, how much it will cost to run this kind of network and "*will usage of it advocate benefits of Blockchain",* as stated before.

Toward Byzos network

As identified during this project, Byzos has big plans for the future as a main data distribution service and as CEO of the company stated his future view: "*Every webshop will pull product related data from Byzos network*". Besides created PoC and benefits identified around it following advices are stated:

- Only Byzos- format of the schema can be saved to the public Ethereum network, lowering the total costs. This stored data in public network would act as "absolute truth"- (or "single-source of truth"), respecting characteristics of Blockchain. Addition to current data struct should also be stored URI for xml-schema endpoint, from where actual product data can be retrieved. This would reduce need of intermediaries and with model of controlling read access of stored schemas, there can be created new business models, like selling access to the 3rd party marketplaces.

  Rest of the schemas can be saved to the permissioned or in to private network, if it is seen needed. Afterall, from Byzos-schema can be reproduced all other needed schema transformations, and from it compare hash when it is necessary.

- To benefit from the concept of decentralization and to improved transparency, data availability and data authenticity decentralised data storage could be used besides Blockchain. For this interesting concepts are IPFS, BigchainDB and Storj. With solution like this, also larger product related data can be stored in decentralized way, like product pictures. With the usage of decentralized data storage could eventually be removed need of the Azure.
- Open discussion about permissioned consortium Blockchain network should be established in the industry and partner companies. With shared Blockchain network, costs of running it can be minimized and benefits of decentralization seen in bigger picture.

Conclusion

Created PoC in this project is a solid proof of how SyncForce can benefit from Blockchain and how it could be implemented to the existing processes. Open question will be solution and expenses related to the type of network. It is advisable that usage of Blockchain is explored in bigger picture, because it can be seen as a main network for distribution and managing published product data. With possible new model and benefits with it possible overhead cost can be cut off.

What comes to the characteristics of Blockchain as a immutable, incorruptible and decentralized data storage can it be seen as a crucial addition in near future to the Byzos and services offered by SyncForce. In the nature of the industry of good manufacturing business, where managing and publishing product data is essential part should service promise be transparent and reliable - this promise is beneficial in eyes of clients and the end customer of the product.