DL-Tags

Grant agreement 688061

Report on use case deployment and trial evaluation

Deliverable ID:	**D3.1**
Deliverable Title:	**Report on use case deployment and trial evaluation**
Revision #:	**1.0**
Dissemination Level:	
Responsible beneficiary:	**UniZG-FER**
Contributing beneficiaries:	
Contractual date of delivery:	
Actual submission date:	**5.12.2018**

Start Date of the Project: 1 January 2016 Duration: 36 Months

Contents

Section 1 - Introduction ***4***

Section 2 - DL-Tags use case ***5***

2.1 Technical solution **5**

2.2 Interactions between the stakeholders **7**

Section 3 - Implementation of DL-Tags use case ***8***

3.1 Producer ..... Error! Bookmark not defined.

3.2 TiS platform **9**

3.3 Magento e-commerce store **11**

3.4 End consumer **12**

Section 4 - Evaluation ***13***

Section 5 - Conclusions ***15***

Document history

Revision	Date	Modification	Authors
v0.1	31.8.2018	ToC	Pavle Skocir
v0.2	15.10.2018	DL-Tags use case (Section 2)	Pavle Skocir
v0.3	21.11.2018	Implementation (Section 3)	Federico Matteo Bencic
v1.0	5.12.2018	Evaluation, Introduction, Conclusion	Pavle Skocir

Section 1 - Introduction

DL-Tags project enhances the existing TagItSmart (TIS) ecosystem with a solution for lifecycle management of products labelled with Smart Tags. DL-Tags relies on distributed ledger technology (DLT) as a mediator in a trustless environment with many stakeholders involved in product handling (e.g. producer, logistic companies, retailers, etc.). The solution is decentralized and privacy preserving. It empowers stakeholders to verify the state of their products of interest and provides insights into all other stakeholders involved in product handling.

This deliverable presents the DL-Tags use case deployment. It demonstrates application of DL-Tags solution in a real-world environment with specific stakeholders. The development of DL-Tags components enabling interaction with the blockchain is presented in *D2.1 DL-Tags implementation*, while this deliverable presents integration of the DL-Tags components with stakeholders' systems. Focus of DL-Tags project is on facilitating product lifecycle management on both Business-to-Business (B2B) and Business-to-Consumer (B2C) market. The former market type includes transactions between businesses (e.g., between a producer and a retailer), while the latter market type includes interactions between a retailer and end consumers [3].

Stakeholders in DL-Tags use case include producer, retailer, and end consumer. The use case considers trading of physical products. An e-commerce store is resolved to be a retailer responsible for obtaining products from producers and offering them to end consumers. E-commerce stores have become widely used for trading not only digital products such as music or books, but also for physical products. According to Statista's Digital Market Outlook, worldwide e-Commerce revenue will grow from 1,2 trillion USD in 2016 to 2,1 trillion USD in 2021 [1]. One of the reasons why consumers tend to shop in offline stores instead of online is because they are concerned that the products could be different than declared [1]. DL-Tags steps into this landscape to devise a solution where product description and product lifecycle can be verified.

The use case deployment is related to brand protection use case defined by TIS. Particularly, DL-Tags solution has been integrated with TagItWine use case for brand protection and anti-counterfeiting in Wine Industry [2]. The integration of TagItWine with DL-Tags facilitates product verification and could provide an insight into the previous stakeholders and all the information about product exchanges between the stakeholders.

Section 2 explains DL-Tags use case in more detail. Implementation of the use case is presented in Section 3. Cost evaluation for interacting with the blockchain is given in Section 4.

Section 2 - DL-Tags use case

A product during its lifecycle is handled by a large number of stakeholders. Firstly produced by its manufacturer, afterwards maneuvered by logistic companies, offered by retailers to interested buyers, and finally delivered to an end consumer. The particular stakeholders of the DL-Tags use case are shown in Figure 1.

DL-Tags use case is envisioned for the brand protection use case defined within TIS project, in cooperation with TagItWine project. The main goal is to show the lifecycle of a bottle of wine offered to the end consumers via e-commerce store. First stakeholder is a producer, i.e., winery in our specific use case. The product is then transferred to a retailer. Magento, an open source e-commerce platform, is deployed as a retailer in our use case. The provider of the Magento e-commerce store is supposed to take the product over from the producer, check its authenticity, and offer it to potential buyers. Consumers using the Magento e-commerce store can find the product online and perform a purchase. When the product is delivered, consumers can check its validity by using the TagItWine mobile application.

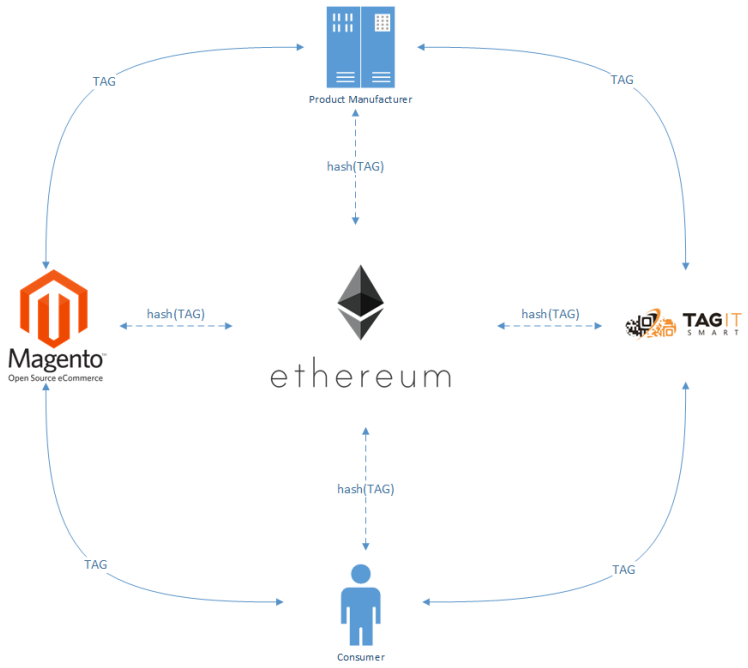

Figure 1 Stakeholders in DL-Tags use case

2.1 Technical solution

The architecture of the DL-Tags solution is elaborated in *D2.1 DL-Tags implementation*. Figure 2 shows components needed to interact with the blockchain. Ethereum is used as a blockchain platform in this project. Each stakeholder (*Platform* in Figure 2) in the product lifecycle needs to deploy its own *DL-Tags Proxy*, a component responsible for handling all the interactions with the blockchain so that the stakeholders do not have to manage blockchain related messages on their own. Additionally, each stakeholder needs to download the blockchain and synchronize with the network. *Blockchain DApp (Distributed Application)* is used to interact with Ethereum blockchain. Each *Platform* needs to use proxy's interface

implemented as JSON RPC (*JavaScript Object Notiation – Remote Procedure Call*) to pass the data that needs to be stored on the blockchain. Additionally, it needs to deploy a message broker implementing AMQP (*Advance Message Queuing Protocol*) for receiving messages from *DL-Tags Proxy* and *Blockchain DApp*.

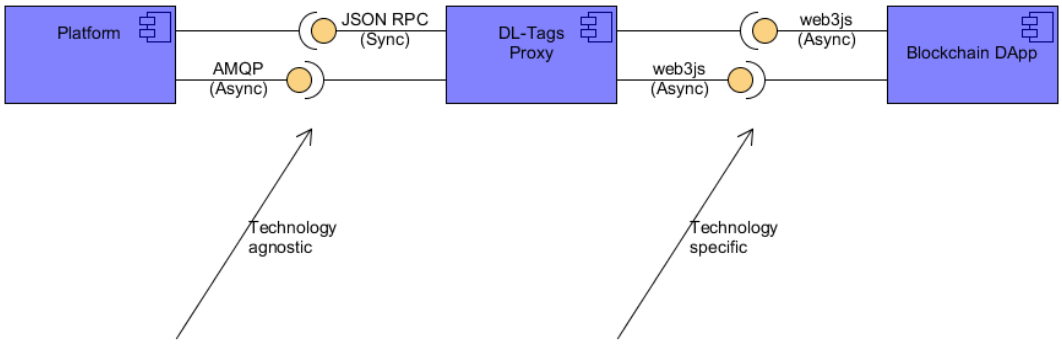

Figure 2 DL-Tags components

All the interactions between stakeholders (e.g. receiving and dispatching a product item) need to be registered on the blockchain. These interactions can be initiated by the blockchain or need to be initiated by the *Platform*. The interactions initiated by the *Platform* include the following:

- Creating a product on the blockchain,
- Adding a new stakeholder to the product,
- Transferring ownership of the product,
- Voting on product validity, and
- Validation check.

The interactions initiated by the blockchain and received by the *Platform* include the following:

- Receiving information about being added as a stakeholder, and
- Receiving information about being made product owner.

Sequence diagrams presenting the aforementioned interactions in more detail are given in *D2.1 DL-Tags implementation* as well. Creating a product on the blockchain needs to be invoked by the product item producer. During this step, TIS system is approached to issue a Smart Tag for the selected product. Upon Smart Tag creation, *DL-Tags Proxy* sets ownership of the product to producer, and adds both TIS system and producer as stakeholders. Producer is notified by blockchain, i.e., its own *DL-Tags Proxy* about being added as stakeholder. This demands a voting message in which producer confirms the Smart Tag validity.

When product owners want to pass a product to another stakeholder, they need to add the new stakeholder to the product. The new stakeholders are notified by the blockchain, and a vote on product validity is requested by the blockchain. After passing a physical product to the new owner, previous owner needs to transfer the ownership. New owner is notified about transferred ownership by the blockchain as well. This process can be repeated for as many stakeholders one product might have.

The last stakeholder who sells the product to the end consumer needs to disown the product. It is not needed that consumers are registered to the blockchain. Disowning of the product means setting product ownership to a *null* value. Consumers can check the validity of the product by contacting blockchain.

2.2 Interactions between the stakeholders

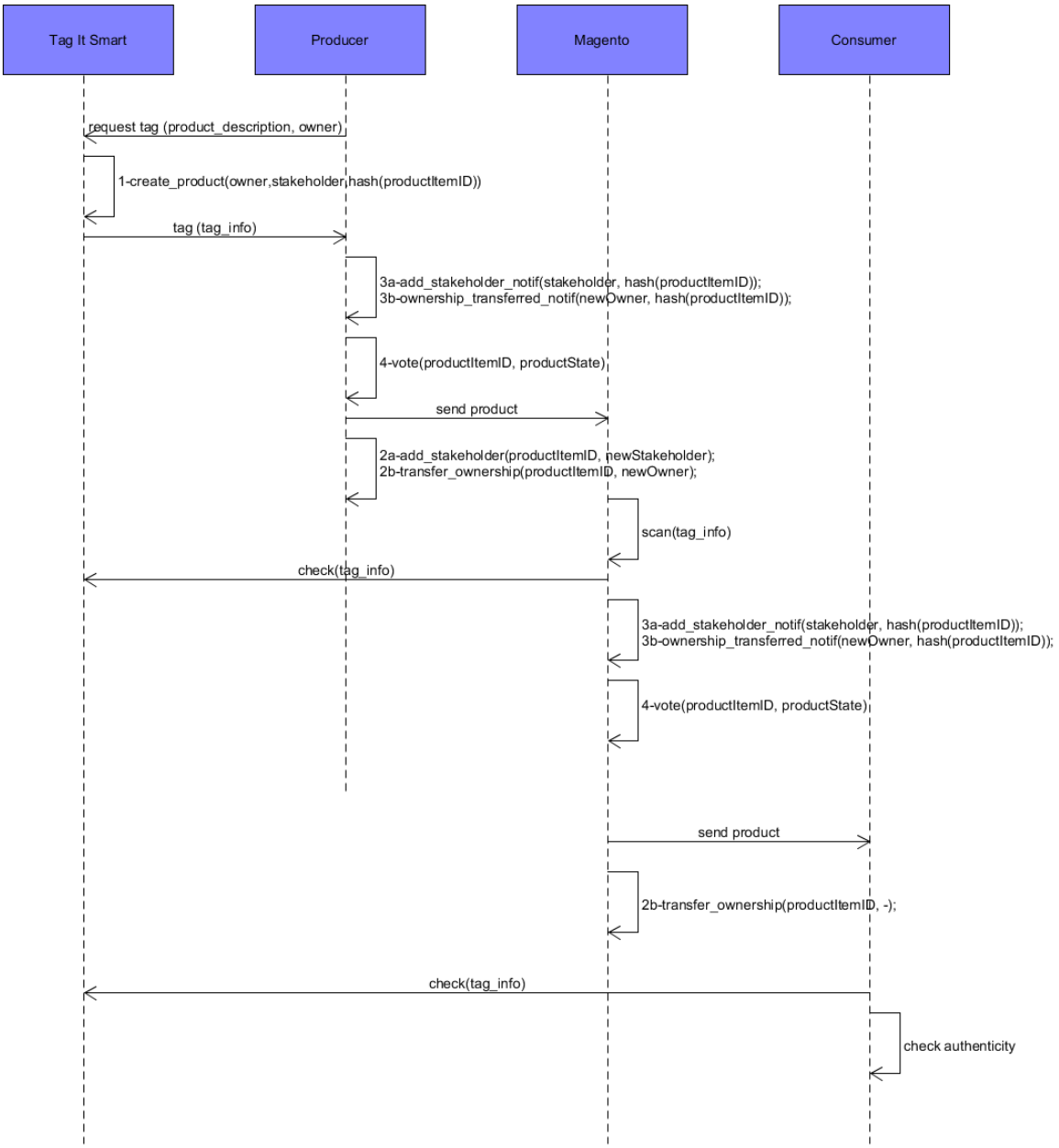

Figure 3 Interactions between stakeholders in the use case

The stakeholders in our use case are shown in Figure 3. Apart from the aforementioned producer and end consumer, there is only one stakeholder, Magento e-commerce platform.

All self-messages in the diagram indicate interactions with the blockchain, i.e. invoking *DL-Tags Proxy's* methods, or receiving messages from *DL-Tags Proxy*.

Message 1-create_product is sent by the *DL-Tags Proxy* representing TIS. Producer is notified by the blockchain of being added as a stakeholder (message 3a-add_stakehodler_notif) and being made the owner of the product (message 3b-ownership_trasferred_notif). After receiving the Smart Tag and checking the Smart Tag info, product owner verifies the Tag and stores the proof on the blockchain (message 4-vote). When producer wants to sell the product via Magento e-commerce store, it needs to send the physical product to Magento store provider, and transfer the product ownership (messages 2a-add_stakeholder and 2b-transfer_ownership). Magento store provider is notified of being made product stakeholder and owner by receiving messages 3a-add_stakehodler_notif and 3b-ownership_trasferred_notif. Upon receiving the physical product, Magento store provider needs to check the validity of the product, and store the decision on the blockchain by voting (message 4-vote). When an end consumer buys the product, Magento store provider needs to disown it (message 2b-transfer_ownership). End consumer receives the product and has the ability to check its validity by using the TagItWine mobile application that interacts with the blockchain.

Section 3 - Implementation of DL-Tags use case

This section describes the functionalities and depicts the developed application for every stakeholder in the use case deployment. Code snippets in this section are written in Javascript.

3.1 Producer

The product owner application initiates Smart Tag creation by contacting *DL-Tags Proxy* of a TIS platform. Sending of the request is shown in Figure 4.

```
let response = await this.clientTiS.requestPromise(
"create", generic.createInvocationParams());
```

Figure 4 Request for Smart Tag creation

After sending the request, producer needs to receive the printed Smart Tag. Additionally, information from the blockchain about being made a new stakeholder and the owner of the product is forwarded as well. Received information from blockchain is shown in Figure 5.

```
[FLOW] TiS proxy creates a product and adds Producer stakeholder
 [x] Received '{"name":"Create","params":{"msg":"Product created at 0x93407711d0dd47a77de865cc2c3ff9531f9dc5b5"}}'
[FLOW] Producer gets properly informed about the creation
 [x] Received '{"name":"StakeholderAdded","params":{"productItemIDHash":"0xf2b0d36a988cb85745073b83d0fa032feec9135b969b19bdebaf3205f4e4d744"}}'
[FLOW] TiS transfers ownership to the producer and is properly informed of success
 [x] Received '{"name":"OwnershipTransferred","params":{"msg":"Ownership transferred to 0xf17f52151ebef6c7334fad080c5704d77216b732"}}'
[FLOW] Producer gets notified about ownership transfer
 [x] Received '{"name":"OwnershipTransferred","params":{"productItemIDHash":"0xf2b0d36a988cb85745073b83d0fa032feec9135b969b19bdebaf3205f4e4d744","previousOwne
4fad080c5704d77216b732"}}'
```

Figure 5 Receiving information about being added as a new stakeholder and product owner

Producer sends the product to Magento store provider to place it online and make it available to end consumers. Along with sending the physical product, product owner needs to add Magento store provider as a new stakeholder and transfer ownership to this new stakeholder, as shown in Figure 6.

```
console.log("[FLOW] Producer adds Magento as stakeholder and is informed adding is successful");
expected = generic.stakeholderAddedSuccessfulInvocationResponse();
response = await this.clientProducer.requestPromise("addStakeholder", generic.addStakeholderInvocationParamsBuilder(generic.magentoBCAddress));
response.result.should.equal(0);

msg = await generic.getFromQueue(this.queueProducer);
msg.name.should.equal(expected.name);
match = expected.params.regex.test(msg.params.msg);
match.should.equal(true);

console.log("[FLOW] Product is deemed not valid after new stakeholder was added");
expected = await this.clientTiS.requestPromise("valid", generic.validInvocationParamsBuilder());
expected.result.should.be.equal(false);

console.log("[FLOW] Magento is informed he is added as stakeholder");
expected = generic.stakeholderAddedEvent();
msg = await generic.getFromQueue(this.queueMagento);
msg.name.should.equal(expected.name);
msg.params.productItemIDHash.should.equal(expected.params.productItemIDHash);

console.log("[FLOW] Producer transfers ownership to Magento and is properly informed of success");
expected = generic.transferOwnershipEventBuilder(generic.tisBCAddress, generic.producerBCAddress);
response = await this.clientProducer.requestPromise("transferOwnership", generic.transferOwnershipInvocationParamsBuilder(generic.magentoBCAddress));
response.result.should.equal(0);

expected = generic.ownershipTransferredSuccessfulInvocationResponse;
msg = await generic.getFromQueue(this.queueProducer);
msg.name.should.equal(expected.name);
match = expected.params.regex.test(msg.params.msg);
match.should.equal(true);
```

Figure 6 Adding a new stakeholder and transferring ownership

3.2 TiS platform

Interactions with the TiS platform are initiated upon Smart Tag creation request invoked by a producer and shown in Figure 4. The detailed interaction is introduced in *D2.1 DL-Tags implementation* and shown in Figure 7. *DL-Tags Proxy* of a producer contacts a *DL-Tags Proxy* of TiS. The latter *Proxy* sends API request to TiS platform. Afterwards, the same *DL-Tags Proxy* invokes product creation on the blockchain, as shown in Figure 8.

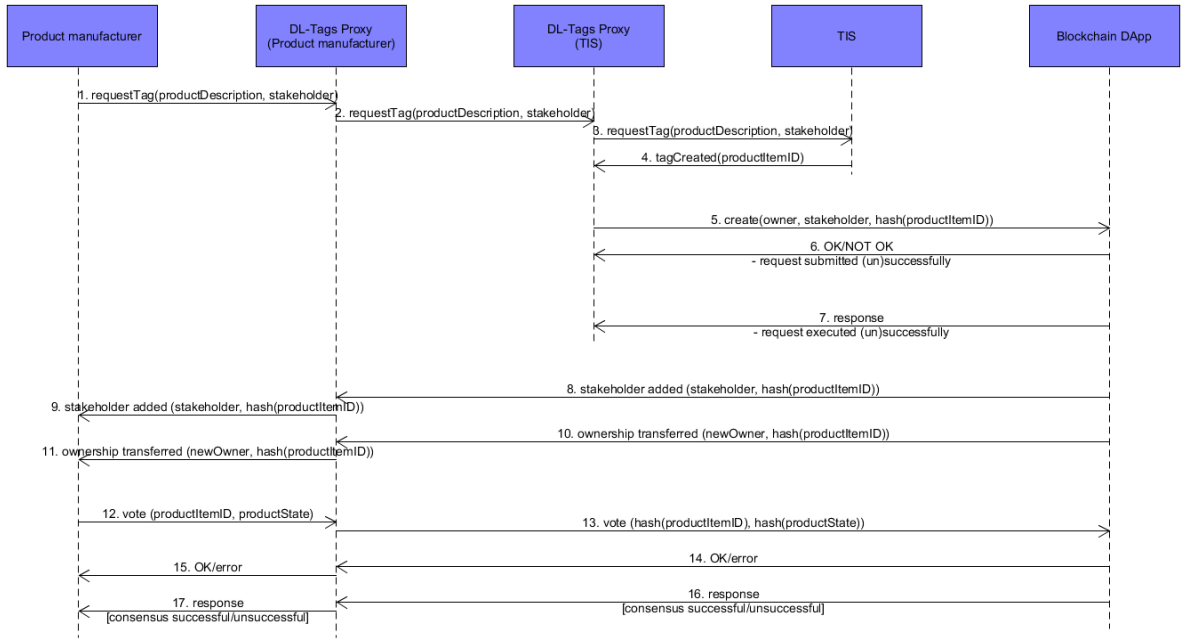

Figure 7 Interaction with TiS platform

```
console.log("[FLOW] TiS proxy creates a product and adds Producer stakeholder");
let expected = generic.createSuccessfulInvocationResponse;
let response = await this.clientTiS.requestPromise("create", generic.createInvocationParams());
response.result.should.equal(0);

let msg = await generic.getFromQueue(this.queueTiS);
msg.name.should.equal(expected.name);
let match = expected.params.regex.test(msg.params.msg);
match.should.equal(true);

console.log("[FLOW] Producer gets properly informed about the creation");
expected = generic.stakeholderAddedEvent();
msg = await generic.getFromQueue(this.queueProducer);
msg.name.should.equal(expected.name);
msg.params.productItemIDHash.should.equal(expected.params.productItemIDHash);

console.log("[FLOW] TiS transfers ownership to the producer and is properly informed of success");
expected = generic.transferOwnershipEventBuilder(generic.tisBCAddress, generic.producerBCAddress);
response = await this.clientTiS.requestPromise("transferOwnership", generic.transferOwnershipInvocationParamsBuilder(generic.producerBCAddress));
response.result.should.equal(0);

expected = generic.ownershipTransferredSuccessfulInvocationResponse;
msg = await generic.getFromQueue(this.queueTiS);
msg.name.should.equal(expected.name);
match = expected.params.regex.test(msg.params.msg);
match.should.equal(true);

console.log("[FLOW] Producer gets notified about ownership transfer");
expected = generic.transferOwnershipEventBuilder(generic.tisBCAddress, generic.producerBCAddress);

msg = await generic.getFromQueue(this.queueProducer);
msg.name.should.equal(expected.name);
msg.params.productItemIDHash.should.equal(expected.params.productItemIDHash);
msg.params.previousOwner.should.equal(expected.params.previousOwner);
msg.params.newOwner.should.equal(expected.params.newOwner);
```

Figure 8 Request for Smart Tag creation

The stakeholder who invoked Smart Tag creation does not need to be aware of the underlying messages exchanged between *DL-Tags Proxies* of the invoking stakeholder and TIS, nor the details of the interaction with TIS for Smart Tag creation. After generation of Smart Tag and product creation on blockchain, producer is notified as shown in Figure 5.

3.3 Magento e-commerce store

Magento e-commerce store provider receives information about the product transferred by the producer through blockchain. The interface through which all information about received products is visible is shown in Figure 9.

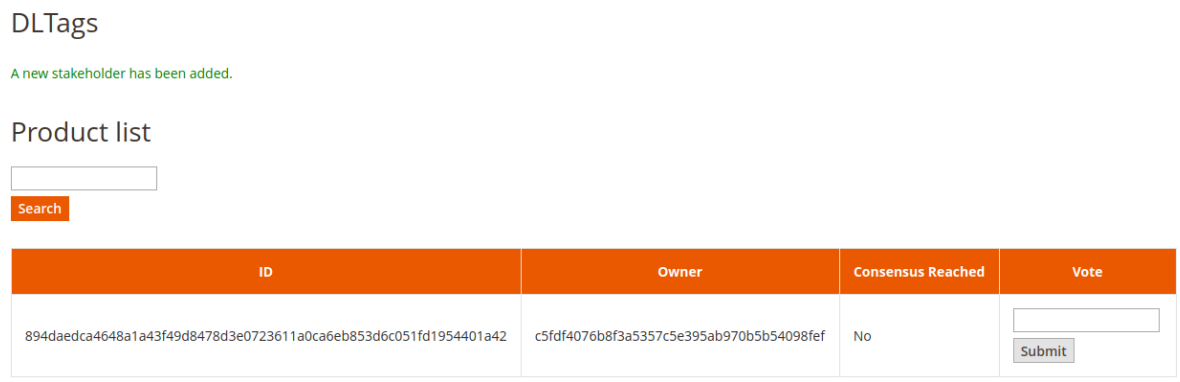

Figure 9 Available products for store inclusion

When the product physically arrives, Magento store provider needs to confirm its validity, i.e., that the Smart Tag is the same as recorded on blockchain. Smart Tag can be scanned by using TagItWine mobile app. Afterwards, Magento store provider enters the code to the textbox in Figure 9. If everything is ok, a consensus reached message is shown, as in Figure 10.

Figure 10 Voting result

Only after successful voting is the product ready to be offered to consumers. The details about Magento e-commerce platform configuration and its modules are given in *D2.2 Magento integration*. After a consumer buys a certain product, Magento store provider needs to disown it. This procedure is shown in Figure 11. The disowned product signifies it has been sold to the end consumer. The interface enabling product disowning is shown in Figure 12.

```
await promptly.prompt("Press any key for Magento to disown the product", {
  default: "y"
});

await this.clientMagento.requestPromise("transferOwnership", generic.transferOwnershipInvocationParamsBuilder(generic.disownedBCAddress));

await promptly.prompt("Press any key for Magento to disown the product again", {
  default: "y"
});
await this.clientMagento.requestPromise("transferOwnership", generic.transferOwnershipInvocationParamsBuilder(generic.disownedBCAddress));
```

Figure 11 Disowning of the product - code

```
Press any key for Magento to disown the product
Press any key for Magento to disown the product again
Press any key to tear down network
proxy.stakeholder3      | info: Ownership transferred to 0x0000000000000000000000000000000000000000
proxy.stakeholder3      | error: Could not transfer ownership
[1] 0:npm*                                                                    "matteoVM" 11:22 05-Pro-18
```

Figure 12 Disowning of the product - interface

3.4 End consumer

To alleviate each end consumer of a need to register to blockchain, end consumers are not added as a stakeholder of the product. Upon receiving the product, end consumers can check their validity. TagItWine mobile application is used and integrated with *DL-Tags Proxy* to enable this feature. User can scan the Smart Tag and receive product information from TiS system, along with verification information from DL-Tags. Successful verification is shown in Figure 13, while unsuccessful verification is show in Figure 14. Unsuccessful verification signifies that the Smart Tag has been manipulated (e.g. issued by a different manufacturer or changed/duplicated by Magento store provider).

Authentication result

Figure 13 Successful verification

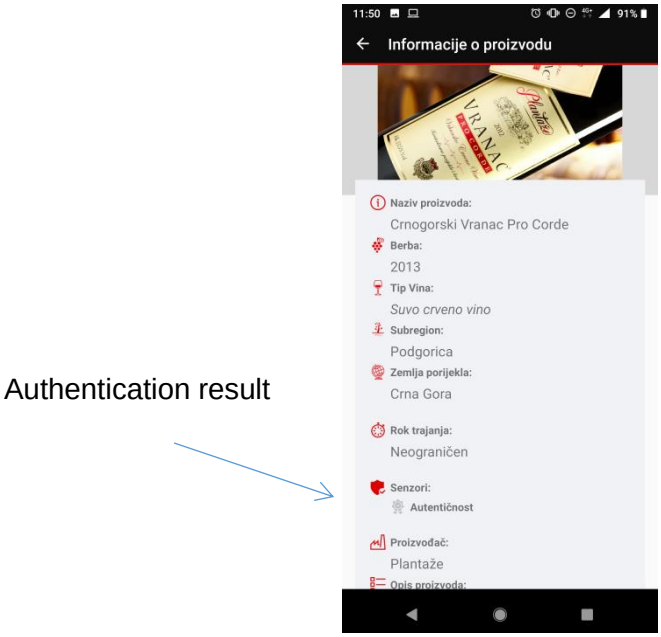

Figure 14 Unsuccessful verification

Section 4 - Evaluation

This section presents cost approximations for invocation of methods defined in DL-Tags solution. The costs are declared in ether and in USD. An operation on top of the Ethereum blockchain is paid in *gas*. A stakeholder can declare how much *wei* (1 ether = 10^{18} wei) is he willing to pay per unit of *gas* for a specific transaction. Depending on the offered amount of *wei* for a transaction, a miner might decide to include that transaction in a block. Putting it simply, the more a stakeholder pays, the more probable it is that his transaction will be included in a block faster. The following values in Table 2 and Table 3 are calculated using the average gas price that users were willing to pay for a transaction in a given period. Herein, we assume that an average gas price implies below 400 seconds block inclusion time. By using the ratio between any fiat currency (USD, EUR, etc.) and ether at that given period we can calculate how much did transaction actually cost. The number of *gas* for each method defined within DL-Tags solution is shown in Table 1.

Table 1 Gas used for method invocation

Method	Gas used
#addStakeholder	63599
#transferOwnership	14649
#vote	89603
#create	1478198
Initial contract deployment	862641

Table 2 shows the cost of each method invocation in ether, while Table 3 shows the cost of each method invocation in USD.

Table 2 Method invocation price in ether[1]

Method/ AVG gas price [GWei]	#addStakeholder	#transferOwnership	#vote	#create	Initial contract deployment	Date
53.86052733	0.003425476	0.000789003	0.0048	0.0796	0.046462299	(15th January 2018)
14.35316497	0.000912847	0.00021026	0.0013	0.0212	0.012381629	(15th March 2018)
48.21618873	0.003066501	0.000706319	0.0043	0.0713	0.041593261	(15th July 2018)
13.26179942	0.000843437	0.000194272	0.0012	0.0196	0.011440172	(15th September 2018)
18.47472879	0.001174974	0.000270636	0.0017	0.0273	0.015937059	(15th November 2018)

Table 3 Method invocation price in USD[2]

Method/ ETH vs. $	#addStakeholder	#transferOwnership	#vote	#create	*Initial contract deployment*	Date
1359	4.655221446	1.072254893	6.5586	108.2	63.14226455	(15th January 2018)
613	0.559575174	0.128889082	0.7884	13.006	7.589938322	(15th March 2018)
433	1.3277951	0.305836105	1.8707	30.861	18.00988212	(15th July 2018)
208	0.175434934	0.040408597	0.2472	4.0775	2.379555758	(15th September 2018)
182	0.001174974	0.049255807	0.3013	4.9703	2.90054465	(15th November 2018)

[1] Historical gas price: https://etherscan.io/chart/gasprice
[2] Historical conversion ether/USD: https://ethereumprice.org/

Section 5 - Conclusions

This deliverable shows use case deployment of the DL-Tags solution. Use case is related to TIS-defined brand protection use case and TagItWine project. The involved stakeholders are producer (i.e. winery), Magento e-commerce store and end consumer. Producer uses a Command Line Interface to interact with TIS and blockchain. Magento store provider uses the defined Magento interface for blockchain interaction, while end consumer uses mobile TagItWine application which is integrated with the blockchain. All stakeholders are using *DL-Tags Proxy* component to interact with the blockchain, as defined in the system architecture.

Any other stakeholder wanting to adopt the solution needs to run its own version of *DL-Tags Proxy,* register to blockchain, invoke methods specified in *DL-Tags Proxy* interface when needed, and run AMQP broker to receive blockchain related data. Alongside, they need to download the blockchain and synchronize with the network.

DL-Tags solution is distributed, without the central node storing product exchange-related information. As such, it could spur adoption of lifecycle management solutions since stakeholders can be reluctant to share their business-related information to third parties that are likely to be required in centralized solutions.

Section 6 - References

[1] Statista: “Offline vs. Online Retail: Development – Comparisons – Consumer View”, Whitepeper, 2017

[2] TagItWine project, http://www.tagitwine.me/

[3] S. Le and Y. Yan, "The integration of B2B and B2C e-commerce mode," 2011 IEEE 3rd International Conference on Communication Software and Networks, Xi'an, 2011, pp. 712-714.