Ref. Ares(2017)1944989 - 12/04/2017

TagItSmart!
Smart Tags driven service platform for enabling ecosystems of connected objects

Grant agreement 688061

Initial Enablers for SmartTags

Deliverable ID:	**D 2.2**
Deliverable Title:	Initial Enablers for SmartTags
Revision #:	**1.0**
Dissemination Level:	**Public**
Responsible beneficiary:	UPC
Contributing beneficiaries:	UPC, UNIS, EVT, UNIPD, VTT, TFE, DNET, DONDE, PICOM, DURST, UNILEVER, UNIVEREXPORT
Contractual date of delivery:	31.3.2017
Actual submission date:	12.4.2017

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

Contents

Section 1 - Introduction ***3***

Section 2 - FC Enablers ***4***

2.1 SmartTag Encoding **4**
2.1.1 SmartTags 4
2.1.2 Physical limitations 5
2.1.3 Secure Encoding 6
2.1.4 Sensor encoding 9

2.2 Identity Management **12**

2.3 Producing of Physical Tags **14**
2.3.1 Printed Tags 14
2.3.2 NFC Tags 15

Section 3 - Semantic model for SmartTags ***16***
3.1.1 SmartTag 'types' definition 19
3.1.2 Semantic Model extension for SmartTag combinations 20

Section 4 - SmartTags in Use Cases ***22***

4.1 Digital Product **22**
4.1.1 Semantic model for SmartTags used in this use case 22
4.1.2 Encoding of sensor 22

4.2 Lifecycle management **22**
4.2.1 Semantic model for SmartTags used in this use case 23
4.2.2 Time-temperature indicator: coding 25
4.2.3 NFC sensor: coding 25

4.3 Brand protection **25**
4.3.1 Semantic Model for SmartTags used in this use case 25
4.3.2 Encoding of sensor 26

4.4 Dynamic pricing **27**
4.4.1 Semantic Model for SmartTags used in this use case 27
4.4.2 Encoding of sensor 28

4.5 Home Services **28**
4.5.1 Semantic model for SmartTags used in this use case 28
4.5.2 Encoding of sensor 30

Section 5 - Conclusions ***32***

Section 6 - References ***33***

Section 1 - Introduction

This deliverable report the initial description of the required SmartTags/Functional Codes (FCs) enablers and the initial solution adopted to manage FCs features, identity and encoding/encryption of information. Final version of semantic model addressing enablers features is also presented.

Physical limitations of FCs are very important factors. Size of the code and reading angle must be considered when generating a code. Hardware of reader devices sets limitations on how complicated codes can be decoded and decrypted fast enough. During this project, partners must find a balance between decoding and usability. Guideline for modern applications is *Identity management* and this is a basic component of the enablers. TagItSmart codes will have a global identity, which is defined during T2.2. Every physical code corresponds to a Virtual Entity (VE), which is key to platform services. *Secure Encoding* takes care that codes cannot be faked. Using visual cryptography, an encoding mechanism is developed further to match security levels of modern IoT service.

In the TagItSmart project we utilize different kinds of methods to generate Functional Codes. We use 2-*D barcodes*, *NFC tags* and *Image based tags (requiring image recognition)*. All these methods require little bit different types of encoding. In the project, we also utilize combination of these methods in some use cases.

In WP2 we have evaluated many different methods to encode *sensor information* into a tag. In printed sensors, printing methods and type of ink are main factors to produce sensor enabled tags. In NFC sensors, we have evaluated how to make optimal sensors for the project needs. In some cases, only combination of a printed tag and a ready-made sensor tag can be a solution. For these cases one must use image recognition to enable reading of a SmartTag.

The rest of the document is structured as follows: Section 2 will cover encoding mechanisms, taking into account physical limitations in sensor codes, identity management and printing, as well as overview the process of SmartTag creation based on the corresponding enablers. Section 3 presents the updates to the SmartTags semantic model, originally presented in D2.1, to account for further work and developments in the project since the D2.1 release. Section 4 presents examples of the SmartTag model instantiations for the project's use cases based on the particularities of the SmartTags used in each of them. Finally, Section 5 concludes this document.

Section 2 - FC Enablers

In this Section, we present the FC enablers that have been developed so far in the context of the TagItSmart process to assist in the overall process of SmartTag creation and deployment.

With reference to the TagItSmart architecture presented in D1.2, these enablers map in the components highlighted in yellow in the figure below.

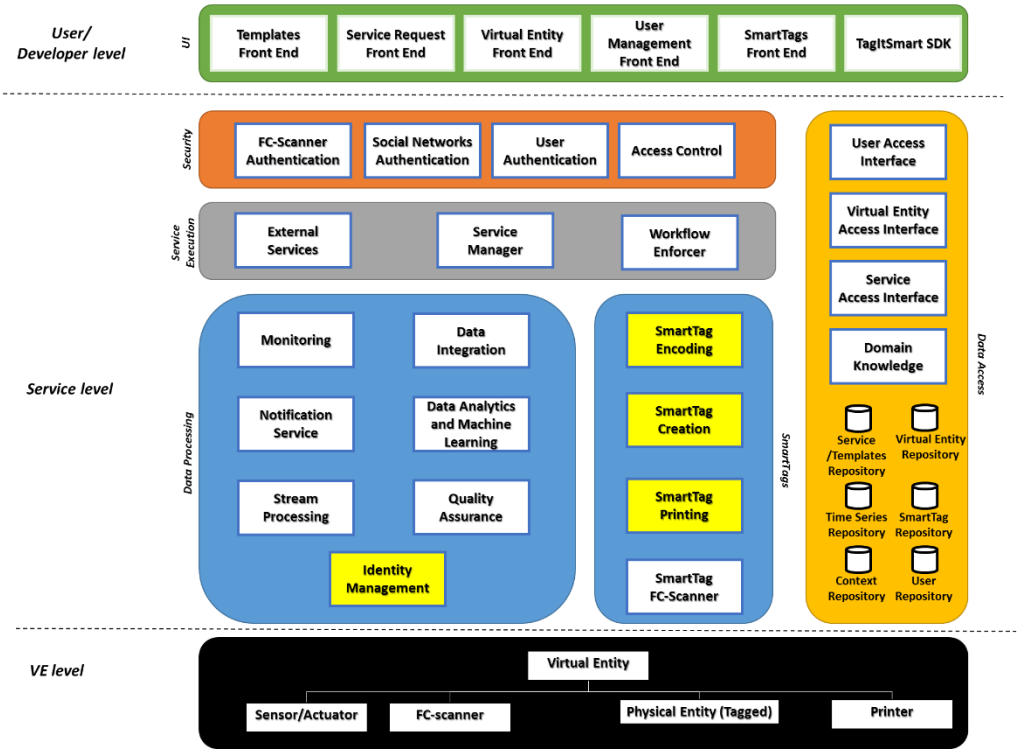


Figure 1: Mapping of FC enablers in this deliverable on the TagItSmart architecture

It is worth noting that components associated with storage, query and discovery of SmartTag information coming from scanning events will be also developed in the context of WP2 but will be presented in detail in later WP2 deliverables.

2.1 SmartTag Encoding

2.1.1 SmartTags

SmartTags are the key element of TagItSmart applications, as they provide innovative sensing capabilities to tags on products. A SmartTag will therefore have to encode at least two pieces of information, the identity of the item and the sensor state. The FC-Scanner will have to be able to read and check both pieces of information.

Encoding strategies are then needed to define how the information about the identity and the sensing capabilities are going to be represented in the SmartTag. The encoding strategy will impact the SmartTag FC-Scanner to be used with them.

Currently, already existing intelligent tags focus mainly on item identification at the product level and there exist different encoding mechanisms:

- *Barcodes*: most wide spread technology, embedded in almost every product in the market (1D Barcodes) and progressively including more sophisticated ones (2D codes, such as QR-Codes and Data Matrix codes). This mechanism encodes the information based on certain geometrical figures and the spaces between them. An optical reader is needed to decode the values.

- *Electronic Tags*: by using small integrated circuits, these tags can store information in a digital format that can be read by a compatible scanner. The most used ones, with varying underlying communication technology are NFC/RFID.

- Another option to implement SmartTags is *image based tags (Labels)*. When smartphones started to include cameras, they started to be leveraged to recognize objects. In this case, the information is encoded in any arbitrary image that represents either the identity of the object or a certain sensed value. This option is still under research. Both encoding and decoding methods needs to be developed further for field trials.

SmartTag Encoding enablers need to provide the mechanisms to integrate the sensor values with the identity of the things to be tagged at the item level, not only at the product level. The process of adding the sensor information to the SmartTags can be tackled in different ways (each strategy will affect how the FC-Scanner is implemented):

- Define a new encoding mechanism that considers the SmartTags, based on the particularities of the ink and NFC technology, resulting in the creation of a new standard. This approach is out of the scope of the project, as it is very intrusive and this will limit the adoption and impact of it.

- Integrate the sensing capabilities of the SmartTags within existing encoding mechanisms. This approach guarantees that the existing encoding standards can be still used but extended with the sensing capabilities. Depending on the original encoding mechanism, this can be achieved in different ways, as it will be shown in the different use cases of TagItSmart. Generally speaking, the objective is to design the SmartTag in a way that the code changes either shape or colour but it is still a valid code in the selected standard.

- Combined mechanisms: this includes any combination of mechanisms to separate the encoding of the identity and the sensing, for example, using a QR code for the identity, which is static and a nearby changing image for the sensor representation, or even two QR codes.

2.1.2 Physical limitations

Due to the different parameters and conditions in which an FC-Scanner is used there are different physical limitations that might occur for the device that will influence scanner performance.

- **Scanning angle:** the angle of scanning may influence the process of scanning in terms of time required to scan the SmartTag and in terms of accuracy (process could

completely fail if the angle and distance are not in the usual range of utilization). Moderate accuracy and performance results are achieved when the SmartTag is scanned directly holding the device parallel to the surface on which the tag is attached.

- **QR code size and scanning distance:** the tag size and scanning distance may influence the scanning process in terms of time required to scan the product and in terms of accuracy (process could completely fail if the QR code size and distance are not accommodated to the usual range). Moderate accuracy and performance results are achieved when the distance of scanning is directly proportional (is growing) with the size of the tag; e.g. during the tests presented in D5.2 [1], for the tag size of 4x4cm the best accuracy is achieved when the tag is scanned from approximately 30cm distance, but this also depends on the FC-scanner hardware (camera quality).
- **Data Matrix code size and scanning distance:** Data Matrix has similar restrictions as a QR code, but the size of code is different. Now (M15) we are using 1.2 cm X 1,8 cm size codes. At the end of this project target size is 1.0cm x 1.0 cm size of code. Reading distance can be shorter than QR code (5 cm). Relying on using optical scanning, the quality of the FC-scanner camera has big influence in the results.
- **Device's hardware:** the quality of the FC-scanner's camera and processing power may influence the scanning process in term of the time required to scan the tag and in terms of the accuracy (process could completely fail if the quality of the device's camera is low); e.g. during the tests presented in D5.2 [1] on device with lower processing capabilities the scanning process took significantly more time.

2.1.3 Secure Encoding

As the usage of QR codes proliferates, multiple security problems were found and exploited by malicious users. We describe the different possible security attack vectors and proposed solutions in the realm of QR codes. QR code could be modified in such a way that a user could be misdirected to a wrong URL, i.e., an attacker can modify the colour of specific modules (a module is a dot that makes up QR code) to modify the encoded content. A good approach to tackle the problem is to derive a checksum (i.e., use a hash function such as SHA-512) and encode the information along with the content, but it would consume the space available for the content. The problem can be even greatly intensified if an attacker modifies the QR code to ensure that it conforms to multiple standards (such as, QR code, Aztec code, data matrix, code with non-square pixels), or by hiding a smaller one within a bigger one. This way, two users scanning the same barcode with different phones or apps will receive different content.

To provide secure encoding to the QR code content, visual cryptography can be used. In recent years, visual cryptography along with QR codes is used for the authentication in e-voting. In the process, unique passwords for everyone were encoded into QR codes, then using Visual Secret Sharing (VSS) two shares, S1 and S2 were generated (and sent to the user by post and digitally), so that either of the layer conveys no information, but when they are combined, the QR code of the password is revealed.

There are two domains we are focusing on: 1) Secret Sharing using QR codes; and 2) Attribute based encryption of QR code to provide different content to the user.

1) Secret sharing using QR codes:
In [2], Jun-Chou, et. al., proposed a secret sharing technique using QR tags. The secret, that the participants want to be shared among them, is divided into shadows, by the (k,n) threshold based secret sharing mechanism, and the results are embedded into Quick Response (QR) codes. The secret can be recovered only when the number of shadows is larger than or equal to a threshold, the proposed technique does not need to connect the back-end database

through Internet. Unfortunately, the content of the tag is meaningless, and the shares can be easily obtained by scanning the QR tags with a bar code reader. In [3] Pei-Yu Lin, et. al., suggested a new secret hiding mechanism for QR tag based on the property of QR codes. The new scheme uses the error correction capability to conceal the secret into a cover QR code. Along with the QR version and the error correction level, the designed algorithm conveys larger secret into a cover QR code directly. Besides, the content of the marked QR code is readable, i.e., general browsers can read the QR content from the marked QR code for the sake of reducing the risk of attention, while only the validated receiver can decode and encrypt the secret from the marked QR code. In [4] Pei-Yu Lin extended the previous work, the proposed approach splits the secret conveyed with QR tags in the distribution application, and the system can retrieve the lossless secret when authorized participants cooperate. General browsers can read the original data from the marked QR tag via a bar code reader and this helps reduce the security risk of the secret. To prevent dishonest participants from obtaining the data, the designed sharing scheme can verify cheaters before revealing the shared secret.

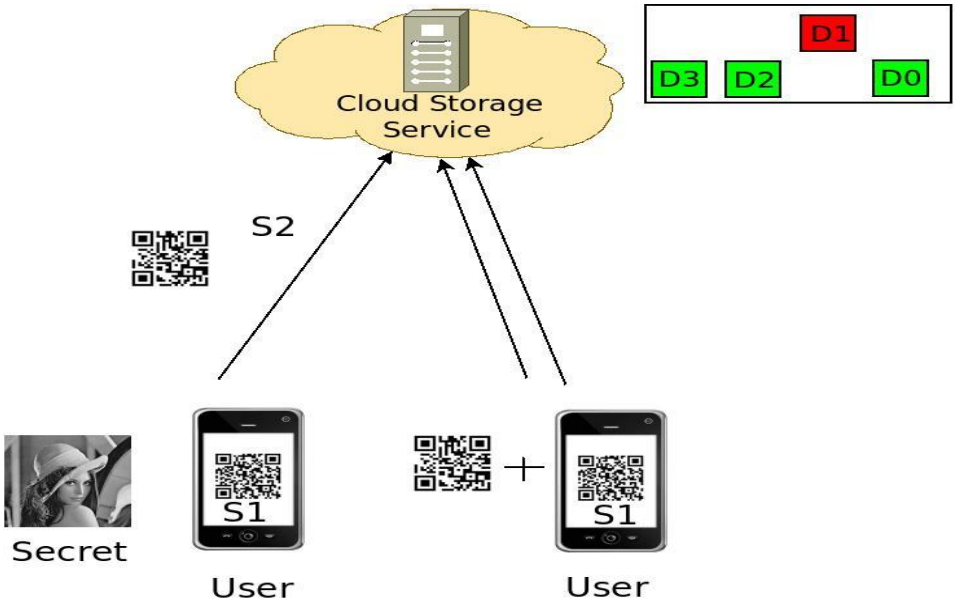


Figure 2: Secret sharing using QR codes

In the TagItSmart platform, Visual crypto will be used to encrypt the QR code. Each user can use images to create visual secrets and share with the backend server, which is (2,2) threshold based secret sharing, i.e., a user generates two shares, S1 and S2 using VSS and sends one of the Share, S2, to the backend server. Every time the user reads a QR code, the QR code is encoded using the other share, S1, and is sent to the server. Server having S2 and the image, generates the secret S1, and uses the shared S1 to extract the QR content (D1 or D2) and responds to the user based on the QR content request (See Figure 2).

2) Attribute-based encryption of QR code to provide different content to the user:
The information provided by QR codes is static, i.e., a particular version of QR code could store only a certain amount of information because of the available number of modules. For example, Version 1 QR code, with L error correction level, can store 152 bits (19 data codewords * 8 modules). The information stored in QR code is also not secured, as anyone using a phone with the available QR reader could extract the information from the QR code. So, there is a need to develop a mechanism to improve the secured information capacity of a

QR code for many users, i.e., to provide different content to the users based on the attributes the user has, for example, identity of the user, the role played, thus improving the amount of information.

Identity-based encryption (IBE), is an important primitive of ID-based cryptography. It is a type of public-key encryption in which the public key of a user is some unique information about the identity of the user (e.g. a user's email address) [5]. This means that a sender who has access to the public parameters of the system can encrypt a message using the text-value of the receiver's name or email address as a key. The receiver obtains its decryption key from a central authority, which needs to be trusted as it generates secret keys for every user. Hence, Identity based cryptography enables any pair of users to communicate securely and to verify each other's' signatures without exchanging private or public keys, without keeping key directories, and without using the services of a third party. Basically, the user chooses his name and network address as his public key instead of generating a random air of public/secret keys and publishing one these keys.

In attribute-based encryption a user will wish to encrypt a document to all users that have a certain set of attribute [6]. For example, an organization encrypts a document to all its employees for a particular tender. In this case, it would encrypt to the identity {"employees-of-XXX-company", "tender-number-YYY"}. Any employee who has an identity that contains all of these attributes could decrypt the document.

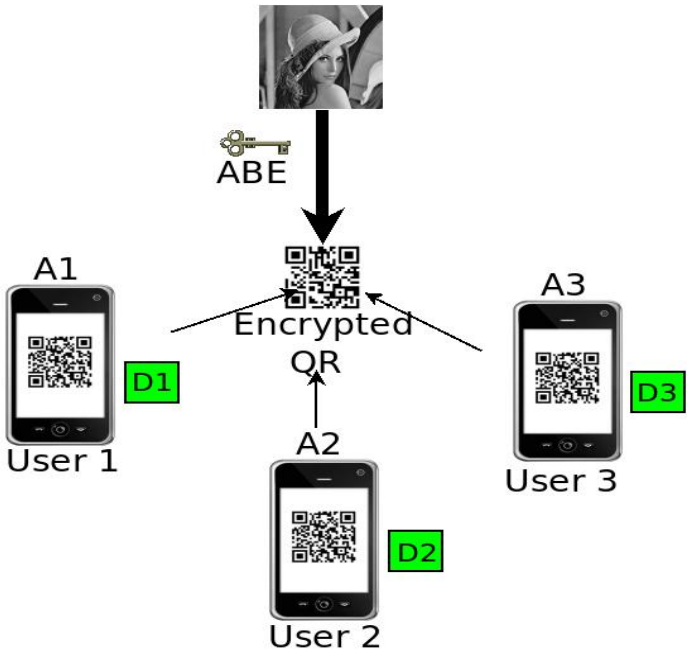


Figure 3: Attribute-based encryption of QR code

The objective of attribute based encryption is to encode the QR code content securely, such that while reading a QR code, the identity along with the attribute of the user will be taken into consideration and different information will be provided to different users based on their identity and attributes. As shown in the figure, while the image is encrypted, attribute based encryption is carried out by considering attributes A1, A2, A3. When users 1, 2 and 3, scan and decode

the encrypted QR based on the attribute they have, i.e., A1, A2 and A3 respectively, the different contents, i.e., D1, D2, and D3 were decrypted.

2.1.4 Sensor encoding

2.1.4.1 QR code

The restriction of printing requirements influence the possible methods that could be used for printing the functional ink as a part of the QR code. More specifically, the process of two-round printing makes it difficult to print the functional ink with sufficient precision in the right place where it is supposed to be printed in the QR code. In addition, the size of area printed with functional ink is also limited as it cannot support printing in small resolutions.

The QR part of SmartTag is encoded as 25-alphanumeric string, which represents 6 different concatenated values:

- Country = 2 (ISO Alpha-2)
- Manufacturer = 5 characters
- Product Line = 3 characters
- Product = 4 characters
- Product Item = 8 characters
- Sensor = 3 characters (these are changing according to change of sensor's observations)

Accordingly, draft proposal for the SmartTag QR code partially printed with functional ink is based on extending the QR code with sensor data using three options for encoding:

- Sensors outside of the QR-Code: the sensors are printed above, below of aside of the QR code

- Sensor inside of the QR-Code: the sensors are printed to shape two forms of QR code both producing quantitative information before and after disappearing of functional ink

- Two QR-Codes interleaved together: the sensors are printed to form two versions of QR code interleaved

2.1.4.2 Data Matrix

Data Matrix FC includes standard DM information and sensor area. Sensor area is overlapping the DM code. Sensor is like a bar in the lower section of the code. Height of this bar is 20 % of

DM code height. Printing of sensor can be done at same time with DM code or it can be done after code printing.

Content of the data matrix contains unique identifier for each item; this identifier is unique within global namespace. Identifier is acquired from TagItSmart compatible server, possibilities for unique identifier are to use 1) identifier itself, 2) short version of identifier or 3) private identifier.

1) Using identifier itself encodes identifier as it is into data matrix; this way only scanners that know how to handle contents are able to perform actions based on the contents. It is also possible to forward this identifier to the backend which then either does some operations based on the identifier or replies to scanner with information about next actions to perform
2) Short version of the identifier is representation where the identifier is shortened by using some algorithms; services like bit.ly work this way.
3) Private identifier is content that is only unique within some defined namespace. These identifiers can be used when the scanner is used as private application within a certain defined group

Figure 4: Data Matrix code with sensor area

Encoding of the data matrix is done as described in the data matrix standard. This means that contents of the data matrix are encoded following standardised rules. Also, data matrix is then printed as any other data matrix. But after this encoding and printing process are done, the sensor is overlayed over the data matrix. Operation can be done vice-versa too, first sensor is printed and then data matrix is created on top of it.

It's also possible to encode data matrix and sensor as whole, if the printing process supports this. In this approach, the data matrix is encoded by following standards, then every white cell within two last rows of data matrix data area is replaced with sensor information

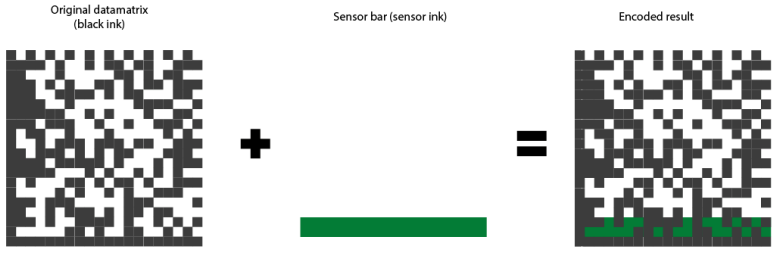


Figure 5: Data Matrix encoding process

If the color of the sensor is light, then scanning specifications that are described in Data matrix-standard are followed. If the color of the sensor is too dark, Data matrix-standards are not followed and code scanning (mainly decoding part) is broken and reading contents from datamatrix is affected by errors. Although it's possible to read data matrix contents, in some cases errors in other sections of the data matrix might completely prevent scanning process.

This option is also usable in case of the QR-Code. But in case of QR-code, bar must not overlap finder pattern.

2.1.4.3 NFC

An NFC tag transmits data according to Thinfilm NFC Barcode Data Format specification. The NFC Barcode is a passive, 128-bit NFC tag operating at 13.56MHz and using a Tag-Talks-First (TTF) protocol. The NFC Barcode operates in a read-only mode to transmit its 128-bit code to NFC-enabled devices, such as phones, tablets, PCs, and set-top boxes.

Contents of the 128-bit NFC Barcode tag are divided into four main sections: the Manufacturer ID, the Data Format identifier, the Data Payload, and the CRC.

Manufacturer ID Field
The 8-bit (1-byte) Manufacturer ID field consists of a start bit and a 7-bit ID. Due to the Tag-Talks-First (TTF) format of the NFC Barcode, it is required that the first bit is always a '1' to serve as a start bit for the NFC controller that is attempting to read the tag. A 7-bit manufacturer ID (based on the least significant 7 bits of the manufacturer IDs specified in the ISO/IEC 7816-6 specification) follows the mandatory '1' start bit. For example, NFC Barcode tags manufactured by Thinfilm under the Kovio brand name (Manufacturer ID 0x37, according to ISO/IEC 7816-6), will have a first byte of 0xB7 (including the '1' start bit).

Data Format Field
The 8-bit (1-byte) data format identifier describes how a reader should interpret the contents of the payload field. The data format identifier contains two sections: Reserved bits and a Data Type Format. The 3-bit Reserved section is set to '000' for the 128-bit NFC Barcode. The 5-bit Data Type Format allows for 32 possible data types, as defined below.

Decimal value	Hexadecimal value	Data Type Format for Fixed (ROM) Data
0	0x00	Reserved for allocation by tag manufacturer
1	0x01	96-bit URL in US-ASCII text, with appended prefix http://www.
2	0x02	96-bit URL in US-ASCII text, with appended prefix https://www.
3	0x03	96-bit URL in US-ASCII text, with appended prefix http://
4	0x04	96-bit URL in US-ASCII text, with appended prefix https://
5	0x05	96-bit binary representation of 96-bit Electronic Product Code (EPC), according to the GS1 EPC Tag Data Standard
6 and above	0x06 and above	Reserved for future use

Data Payload Field
The interpretation of data located in the payload is governed by the contents of the tag's Data Format Field. For the 128-bit NFC Barcode, the payload length is 96 bits (12 bytes). The payload is programmed during manufacturing and therefore is read-only. Contents of the payload depend on state of sensing line. Sensing line can be in two states, either intact (sealed state) or torn (opened state). Data format cannot change depending on a state, i.e. tag will

either transmit two different manufacturers defined IDs for sealed and opened states or two different unique URLs of same type. Tags without a sensing line simply transmit static payload of particular type.

Manufacturer Defined ID (0x00) Payload
Code 0x00 is reserved for allocation by the tag manufacturer. It may, as an example, serve as a globally unique identifier within the manufacturer's tag identification system when combined with the tag's manufacturer ID located in Byte 0.

URL Payload Types (0x01-0x04)
For URL data, the 96-bit payload consists of exactly 12 bytes of information, representing 12 7-bit USASCII characters. For more information, consult Section 3.2.3 "URI Field" of the NFC Forum "URI Record Type Definition, Technical Specification, version 1.0." The specification can be downloaded after accepting the license at http://www.nfc-forum.org/specs/spec_license An optional terminator byte (0xFE) can be used to indicate the end of a URL with a length less than the maximum allowed in the tag payload area. Note that any payload data following the terminator byte and before the CRC is not to be included as part of the URL. Any data following the terminator byte must, however, be readable as part of the tag's complete 128-bit code. The presence of a terminator byte has no impact on the calculation of the CRC.

GS1 Electronic Product Code (EPC) Payload (0x05)
For 96-bit Electronic Product Code (EPC) data, the 96-bit payload consists of a 96-bit binary representation of a 96-bit EPC as defined in the GS1 EPC Tag Data Standard. The latest version can be downloaded directly from GS1 at http://www.gs1.org/gsmp/kc/epcglobal/tds

CRC Field
For the 128-bit NFC Barcode, the final 16 bits (2 bytes) contain a 16-bit CRC for the previous 14 bytes. This CRC is calculated according to the ISO 14443 Type A CRC defined in Appendix B of the ISO 14443 Part 3 specification. Checking this CRC provides a fairly robust method of identifying code transmission errors. For tags containing a combination of fixed (ROM) and dynamic data, the transmitted CRC will change based on the state of the dynamic bits and will automatically correspond to the tag's combined fixed and dynamic data. No change is required to the NFC Controller to support tags containing both fixed and dynamic data.
Note: The initial bit '1' and 16-bit CRC are considered to be part of each NFC Barcode's 128 bit code and should be included in the data transmission from the NFC Controller to the host, even if the CRC is analyzed and verified by the NFC Controller.

2.2 Identity Management

One of the Internet of Things greatest potentials is the promise of scaling to billions of items and devices, if not more. The basic level of identification of an item is to be able to look up its specific properties, usually in terms of description and branding information. In the world, there are currently some widely-adopted identification standards (such as the ones under the GS1 organization [http://www.gs1.eu/] that gives a mechanism to create globally recognized identifiers for products.

However, these identifiers currently are mostly used to recognize products at the SKU level, which leaves two key aspects to transform a product into a digital asset unsolved. On the one hand, how to identify the specific item itself, not only the product and on the other hand, objects connected to Internet will have a unique lifecycle and can be shared among different stakeholders and different systems. How can we fill this gap in identification level and

connectivity? A novel idea, inspired by the Web of Things [https://www.manning.com/books/building-the-web-of-things] is based on the creation of Web Identities for each single item, providing a simple yet powerful standard API to communicate with them, extract information and trigger services.

In TagItSmart terms, we will follow the Web of Things principles to provide Virtual Entities (following the presentation in D1.2, items with SmartTags attached are considered s VEs) with a Web Identity that can be easily shared and integrated in the rest of the components and services. A Web Identity provides a uniquely addressable endpoint through a defined API, that gives access to metadata about the item itself, as well as historical data and any additional service that is provided by the Virtual Entity itself. There are different ways to implement Web Identities, depending on the underlying platform and infrastructure. The next figure shows the basic steps to go from an object to a Virtual Entity with a Web Identity.

Figure 6: Identity Management Enabler process from Object to Web Identity

The Identity Management enabler provides the necessary infrastructure to create and manage the Web Identities of the Virtual Entities. The enabler provides an API for the creation of Web Identities and the associated Virtual Entities. Implementations of this API need to scale and provide certain batch and on-demand creation capabilities to be able to deal with the real-world use cases, for example, as described in the use cases' information flows in *D1.2 Initial TagItSmart! Reference Architecture and API description,* the provisioning of the Web Identities could be done in advance, after the item is well known or even integrating the Identity Management enabler with external systems.

Additionally, due to the diverse nature of the use cases we are implementing in TagItSmart, Web Identities are compatible with pre-existing identification mechanisms, as extensions of the Virtual Entity model. This way the compatibility with existing systems is guaranteed, what is key for integration and adoption.

SmartTags will not have a unique identity per se. SmartTags are thought to have a twofold functionality, one is to identify the item and the second is to provide sensing capabilities on certain properties of that item. Based on this concept, the encoding of the SmartTag needs to combine, as presented in the following sections, the Web Identity of the Virtual Entity and the sensor information.

2.3 Production of SmartTags

For the production of SmartTags based on encoding and identity information, the SmartTag Creation and SmartTag Printing enablers are involved, implementing the workflows presented in Figures 20 and 21 of D2.1.

2.3.1 Printed Tags

For printing functional inks, both analog and digital printing methods are suitable. In analog printing (e.g. flexography, screen printing, offset) a physical master, a printing plate, is required for image reproduction thereby not making analog printing economically feasible for printing customized (individual) 2D bar codes. Analog printing is, however, capable of producing large batches in high speed with static printing layouts. Digital printing methods (e.g. inkjet, electrophotography), on the other hand, are suitable for printing customized, even personalized, 2D codes since the printing layout comes from a digital file without a physical master. There are inkjet printers available designed for printing variable data, such as Data Matrix and QR codes, for coding and marking applications. These printers are suitable for integration with other production and printing lines, and produce variable data on-the-fly. Manufacturers include Domino Printing (www.domino-printing.com) and Videojet (www.videojet.com). TagItSmart project partner Durst manufacturers label inkjet printers also suitable for printing variable data with 2D codes.

In *D2.1 (Semantic model for FunCodes)* commercially available functional inks and indicators have been screened. There are several different ink technologies available for analog printing methods, and those are readily usable with existing analog printing presses. For digital printing the availability is limited to only a few technologies, mainly UV or IR fluorescent inks, not relevant for TagItSmart project's use cases.

There are, however, some limitations related also to functional inks for analog printing. The colours and colour changes available are limited. For example, with thermochromic inks the colour disappears when the temperature is higher than a specific limit. In some cases, it might be preferred to have an appearing ink instead of a disappearing one. Another issue is the difference between inks and indicators. For some cases, such as time-temperature monitoring (product at a particular temperature longer than a particular duration), there are only indicators available - not inks. Monitoring of time-dependent temperature changes is somewhat complicated and requires a logging capability that cannot be achieved with a single ink. This limits the usage scenarios since use of variable data is not possible due to the need to purchase ready indicators from the manufacturers.

To conclude, when selecting the functional performance of the SmartTag, it is important to understand the limitations and capabilities of the functional ink technologies and indicators, and select the functional performance based on their availability and properties. Also when variable data (e.g. individual 2D codes) are required for SmartTag production, a proper selection of the printing methods is required due to the limited availability of the functional ink technologies for digital printing methods. In most cases hybrid solutions (combination of analog

and digital printing methods) should work, where the 2D codes are printed with regular inks with digital printing, and the functional parts with analog printing or assembled from indicators.

2.3.2 NFC Tags

Thinfilm' s PDPS (Printed Dopant Polysilicon Technology) is used to produce the PIC (Printed IC) used in the NFC labels.

The production has the following steps:

a. PIC Production
b. Wafer conversion (PIC bump and dice)
c. Dry inlay assembly
d. Wet inlay conversion

The PICs are currently produced in a sheet-based process where 4 out of 8 layers are printed.

The PDPS (Printed Dopant PolySilicon) PICs for NFC Speedtap and NFC Opensense are produced at Thinfilm Inc in San Jose, California, USA. Thereafter, the sheets are shipped to China where they are converted at third party suppliers. The wafers are diced and thereafter converted to dry inlays and finally into wet inlays.

After the conversion to wet inlays, the remaining steps are to place the labels on the final product. This is typically done by a converter or by the customer. Thinfilm have qualified converter partners to assist with this steps.

In order to test the NFC labels, a test setup is being built in Linköping, Sweden. The test setup will include evaluation of the NFC scanning reliability, the NFC reading distance as well as the NFC scanning response time.

Thinfilm is in the process of scaling up the production of the PICs which is the base of the NFC products. The objective is to invest in roll to roll production equipment with the goal of being able to produce 5 billion PICs annually.

Section 3 - Semantic model for SmartTags

This section details the extensions to the SmartTag model detailed in deliverable D2.1 and presents the first instance of model validation by creating instances of the various smart tags used in the different use cases to be presented in Section 4.

The SmartTag model has been extended to support the validation enablers and service templates. The model extension is in terms of specification of:

1) new kinds of SmartTags: which include a Label class specification, which depicts, for instance, a time-indicator or temperature-time-indicator label, and a Sensor_Area specification, which depicts an area printed below a standard QR code, with functional inks that reacts to changing environmental conditions;
2) combinations of the different modelled kinds of smart tags, namely, datamatrix-based, QR code-based and NFC tags along with possibly available environment, and/or time-reactive labels; and
3) specialized SmartTag 'types', depicting, for instance the different kinds of data matrix-based or QR code tags used in the different TagItSmart UCs. The different types have been defined to mirror the reactions to environmental conditions by the tags used in the UCs, e.g. reactions to temperature thresholds after exposure for a defined period.

The resulting top-level hierarchy of the SmartTags is as shown in Fig. 7 below and the resultant SmartTag model is as depicted in Figure 8.

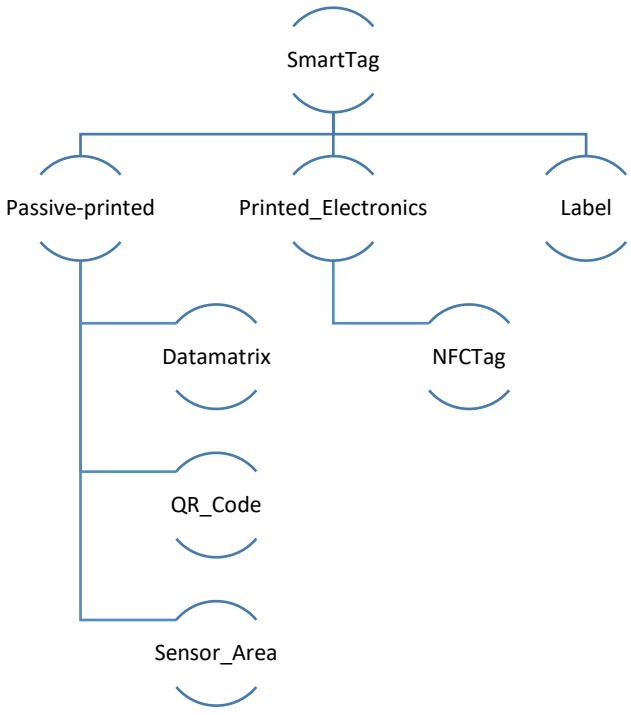


Figure 7: SmartTag hierarchy

The Label class has been defined as a sub-class of the root SmartTag class. In addition to the tag ID, name and size specification (the TagSize class and the corresponding properties have not been shown in Figure 8 due to space restrictions), it also specifies the lifecycle state attached with the label (reversible/irreversible). It has several object properties capturing its characteristics relating to response to specific environmental conditions, for instance,

temperature, time lapse or combination of these, e.g. exposure to a specific temperature for a certain time. This is captured through the *reactsTo* property that links to an instance of the specific environment feature that this label responds to. Like the rest of the SmartTags, the other properties specify the decoding mechanism used, the Measurement recorded by the label and the allowed state changes. The state change specification is the same as applied to functional inks, as detailed in deliverable D2.1. Specifically, it captures the threshold value (if any) at which the state transformation is triggered, the visible (colour) transformation involved and any optional time lapse indications if associated to the state change. Together, these three conditions allow capturing a range of different combinations of the possible labels that can be encountered in the different UCs, including time indicator labels (which only have the time-lapse value of the state change) and temperature-time indicator labels (which have both time-lapse and threshold combined with colour change specifications).

For any SmartTag having some measure of time lapse involved, an additional *activationTime* property has been introduced, which captures the date and time at which the time indicator is activated. This property is associated with the *TimeLapse* class, with *xsd:dateTime* as its range.

The *Sensor_Area* class has been defined to depict a patch printed using functional inks (thermochromic or hydrochromic etc.) to show visible transformations as a response to the corresponding environmental condition, e.g. a sensor area printed using hydrochromic ink reacts to the presence of humidity/water. Therefore, the *Sensor_Area* class has been defined as a sub-class of the Passive-Printed class and inherits all the properties and restrictions of its parent class.

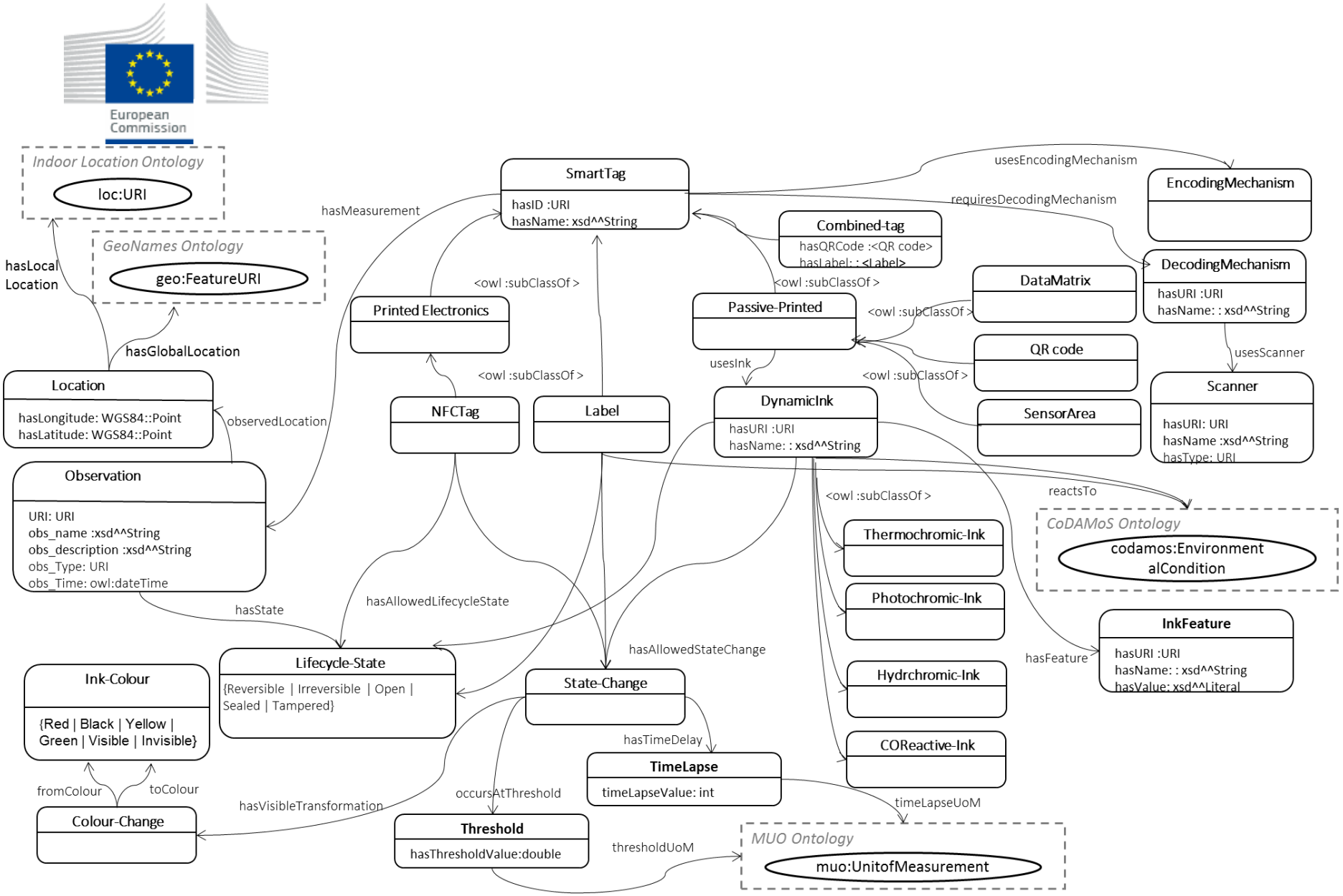


Figure 8: SmartTag ontology

3.1.1 SmartTag 'types' definition

In the semantic model, the different types are modelled by specialization of the different classes by placing appropriate restrictions on the values of the properties of the classes. This is illustrated in the following paragraphs with the example of the digital products use case.
Since the digital products UC makes use of a datamatrix SmartTag, the corresponding semantic concept in the SmartTag model is first specialized with a sub-class, i.e. *DigitalProduct-Datamatrix* class, through the <*owl:subclassOf*> hierarchical representation, as represented in (1) below:

$$DigitalProduct_DataMatrix \sqsubseteq DataMatrix \quad (1)$$

The *DigitalProduct-Datamatrix* class inherits all the properties of its parent *Datamatrix* class, viz. tagID, tagName and the object properties linking it to the encoding and decoding mechanisms classes as well as that linking it to the Observation class.

To represent the special type of digital ink used in the digital products UC, which is a type of thermochromic ink, the corresponding Thermochromic_Ink class is specialised to a *DigitalProduct-ThermochromicInk* class to capture its unique properties and allowed values of state change. The new sub-class is associated to the *DigitalProduct-Datamatrix* class by restricting the *DigitalProduct-Datamatrix* class' *usesInk* property to instances of only the *DigitalProduct-ThermochromicInk* class, as shown in (2):

$$DigitalProduct - DataMatrix.usesInk \ni DigitalProduct - ThermochromicInk \quad (2)$$

As the thermochromic ink employed in the UC is a reversible ink, this is depicted in the semantic model by placing a 'hasValue' (∋) restriction on the *hasAllowedLifecycleState* property of the *DigitalProduct-ThermochromicInk* class:

$$DigitalProduct - ThermochromicInk.\text{hasAllowedLifecycleState} \ni \text{Reversible} \quad (3)$$

The permitted state change of the thermochromic ink is represented by creating a sub-class of the *StateChange* class, i.e. *DigitalProduct-StateChange* and placing appropriate restrictions on its properties. Specifically, these include restricting its *hasVisibleTransformation* property to only the *DigitalProduct-ColourChange* class, which in turn restricts its from and to colour changes to invisible and green, respectively, as shown in (4):

$$\begin{aligned} DigitalProduct - ColourChange.\text{fromColour} &\ni \text{Green} \\ DigitalProduct - ColourChange.\text{toColour} &\ni \text{Invisible} \end{aligned} \quad (4)$$

Simultaneously, the *DigitalProduct-StateChange* class has its threshold value restricted to the specialised *Threshold_8* class, which specifies that it has an integer threshold value (*hasThresholdValue* property), which is less than or equal to 8:

$$Threshold - 8.hasThresholdValue \ni integer\ [\leq 8] \quad (5)$$

Following the definition of the digital products UC SmartTag type above, an instance of the datamatrix SmartTag is then created, as shown in Fig. 9.

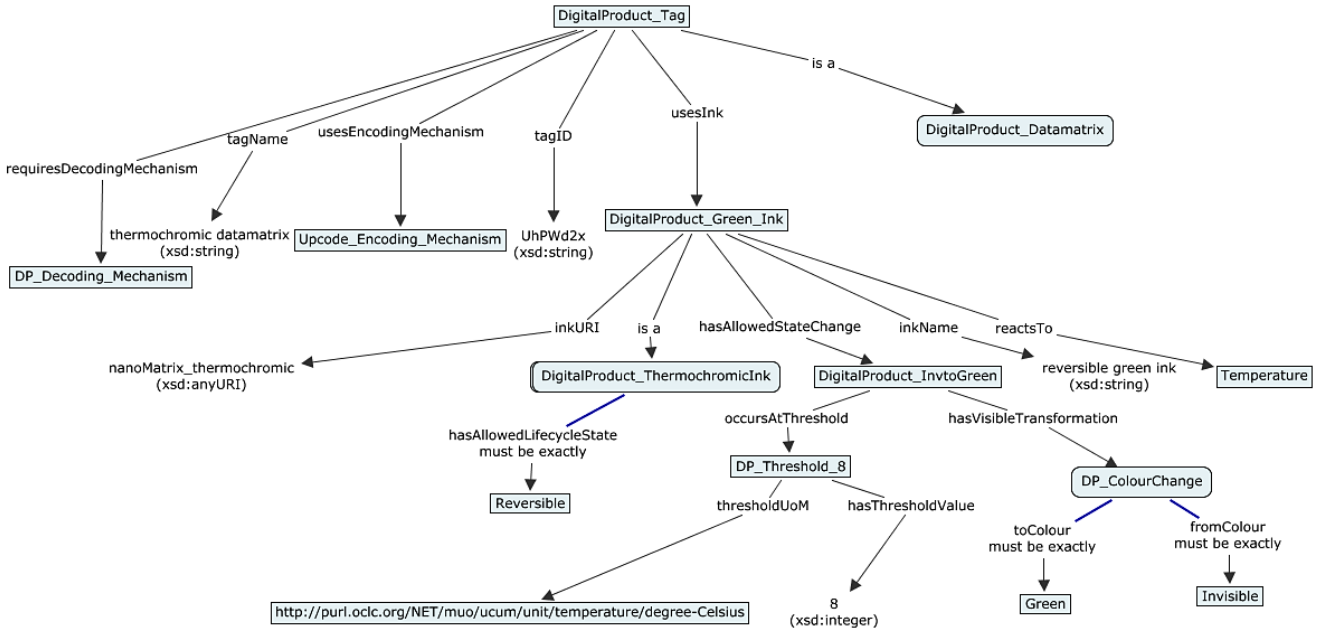


Figure 9: Semantic model of Digital Product use case

3.1.2 Semantic Model extension for SmartTag combinations

An analysis of the UC requirements for the different smart tags that are encountered in these reveals that there are two required combinations: a) combined tags – where two different kinds of SmartTags (e.g. QR-code and Label) are encoded into a single SmartTag label, with the combination having one unique ID but retaining and showcasing the individual characteristics of the constituent kinds of smart tags, b) non-combined tags – where two different kinds of SmartTags are present adjacent to each other on a surface (e.g. on a sticker) but have distinct unique IDs as well as the individual characteristics of the SmartTags involved.

For the needs of this Section we use the LifCycle Management use case as an example.

To model SmartTag combinations, the *Combined_Tag* class has been defined in the semantic model, as shown in Fig. 8. It inherits the foundational properties from its parent SmartTag class concept, which include the *tagID*, *tagName*, *size* and measurement. Additionally, to model the constituent SmartTags, it defines a number object properties: *hasQRCode, hasLabel*, *hasNFCTag* etc. (only the hasQRCode*, hasLabel* properties are shown in Fig. 8 due to space constraints) with the property ranges restricted to instances of the corresponding classes. This enables any combination of the different kinds of SmartTags as have been defined in the SmartTag model.

For the combined QR code, temperature-time indicator (TTI) label SmartTag used for the ice creams, as described in section 4.2, the *QRCode-TTI_Tag* instance is defined as an instance of the *Combined_Tag* class. It instantiates the 2 object properties: *hasQRCode* and *hasLabel*, with the property ranges restricted to instances of the corresponding *QR_Code* and *Label* classes, respectively.

These two object properties allow linking to instances for the constituent QR code and TTI label, thus capturing their individual characteristics. Following the description of the properties of the combined label for ice cream lifecycle management, as outlined in section 4.2, the generated instance of the QRCode-TTI_Tag class is as shown in Fig. 10:

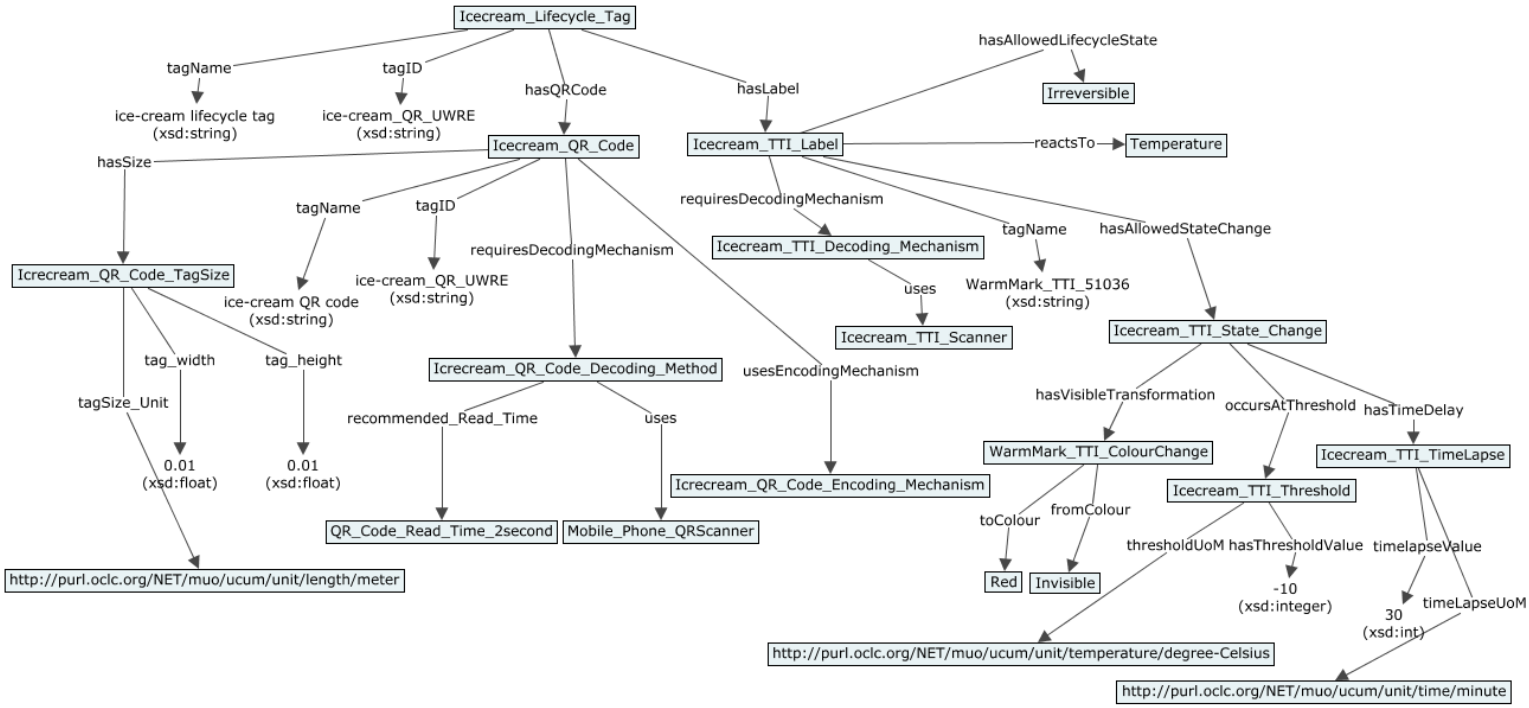


Figure 10. QR code-Time temperature indicator combined label

The QR code part of the combined label supplies the unique ID. The individual characteristics of the QR Code and TTI Label parts are shown with the property values of the corresponding instances. The QR code part includes its ID and tag name, the size of the generated QR code and its associated decoding mechanism, which includes specification of the scanner to be used and the recommended tag read time. The Label instance, the Icecream_TTI_Label, includes the environment feature that it responds to (temperature in this case), a string name specification and the state change. The state change specification captures the important aspects of this combined label, as it specifies the visible colour change to be encountered, the temperature threshold which triggers the colour change and the time lapse involved in the colour change trigger

Section 4 - SmartTags in Use Cases

4.1 Digital Product

In this use case, the used sensor is thermochromic sensor (+8 ◦C). Sensor encoding is very simple, because of this method use standard area for every sensor.

4.1.1 Semantic model for SmartTags used in this use case

The semantic model instantiation for the SmartTags used in this use case was presented in Section 3.1.1, (Fig. 9) where it was used as an example to illustrate how specialization of the different SmartTag ontology classes can lead to different SmartTag types definition.

It is worth noting that using similar reasoning, the semantic model instantiations for the other use cases were also created.

4.1.2 Encoding of sensor

Encoding the sensor in case of digital product is done by applying sensor bar over existing datamatrix tag. Depending on the label printer this also be done during printing of the datamatrix, this way only one printing process needs to be done. This is ongoing process which also involves business aspects so either one described earlier can be used. In both solutions printed sensor bar is almost invisible in temperatures over (+8 ◦C) and under (+8 ◦C) green bar becomes visible.

As the selected approach, doesn't require changing actual data matrix encoding process, only "encoding" is done during printing, this can be illustrated with following figure

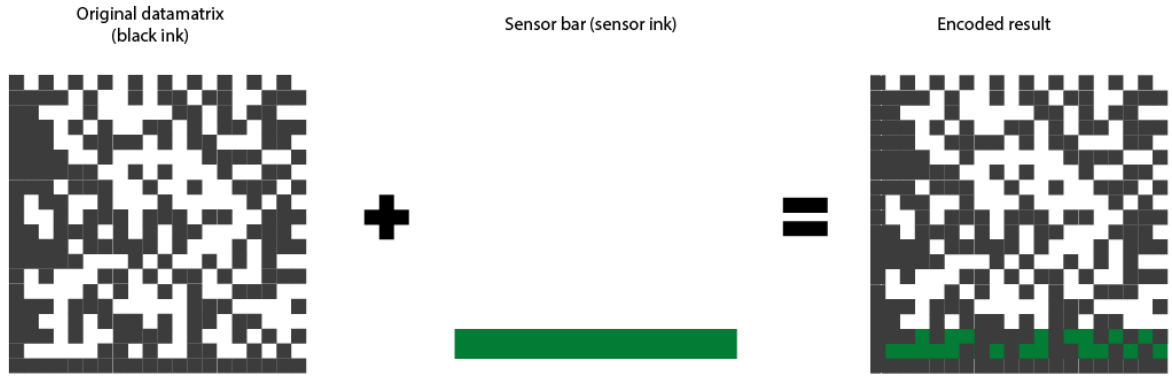


Figure 11: Sensor bar placed over code

4.2 Lifecycle management

In this use case there are different types of Smart Tags:

- **QR code** and **Time-temperature indicator** (combined tag): A combined tag sticked manually in a batch of Unilever ice creams. It will be used to identify each ice cream and tell the end-user if the quality of the product has been compromised:

- **QR code**: provides a unique identifier for each ice cream. The code will be readable by most mobile phone scanners. The desired size of the code is 1.0 cm x 1.0 cm. and should be readable in under 2 seconds.
- **Time-temperature indicator (TTI)**: provides an indicator of the quality of the ice cream based on its cold chain. The indicator will be readable only by scanners that are TagItSmart compatible. The indicator will use irreversible thermochromic ink that will change its colour (red) after exposure to a temperature above -10 C° for more than 30 minutes.If the Time-Temperature Indicator is visible, it will warn the end-user that the quality of the ice cream has been compromised.

- **NFC tag 'sort sensor'**: A widget with an embedded NFC tag that can be sticked to home bins. It will be used to allow end-users automatically download the App or launch the app by just tapping it:
  - **NFC tag 'sort sensor'**: provides a unique identifier for each end-user/home bin. The tag will be readable by mobile phones with NFC readers. The desired size of the tag is 1.0 cm x 1.0 cm. and should be readable in under 2 seconds.
- **QR code** and **NFC tag 'smart street container'** (non-combined tag): A non-combined tag sticked manually on every street container in the pilot (both Smart Tags will be part of a bigger sticker placed in the container that will explain how the system can be used). It will be used to identify each street container and complete the recycling action by the end-user to get rewards. End-users will be able to use just one method (QR code or NFC tag) to make the 'virtual' interaction between their recycling activity and the container.
  - **QR code**: provides a unique identifier for each street container. The code will be readable by most mobile phone scanners. The desired size of the code is 4.0 cm x 4.0 cm. and should be readable in under 2 seconds.
  - **NFC tag 'smart street container'**: provides a unique identifier for each street container. The tag will be readable by mobile phones with NFC readers. The desired size of the tag is 4.0 cm x 4.0 cm. and should be readable in under 2 seconds. The NFC tag must be high temperature proof to allow container cleaning processes.

4.2.1 Semantic model for SmartTags used in this use case

The semantic model instantiation for the combined "QR code and TTI label" SmartTag was already presented in Section 3.1.2. The NFC tags are used as a home bin sort sensor in the lifecycle management UC. As outlined earlier in this, the NFC SmartTag provides a unique ID to the home bin, has features including tag size and requirements related to the decoding mechanism which include specification of the scanner which can be used and the recommended read time. Its corresponding semantic instance is shown in Fig. 12.

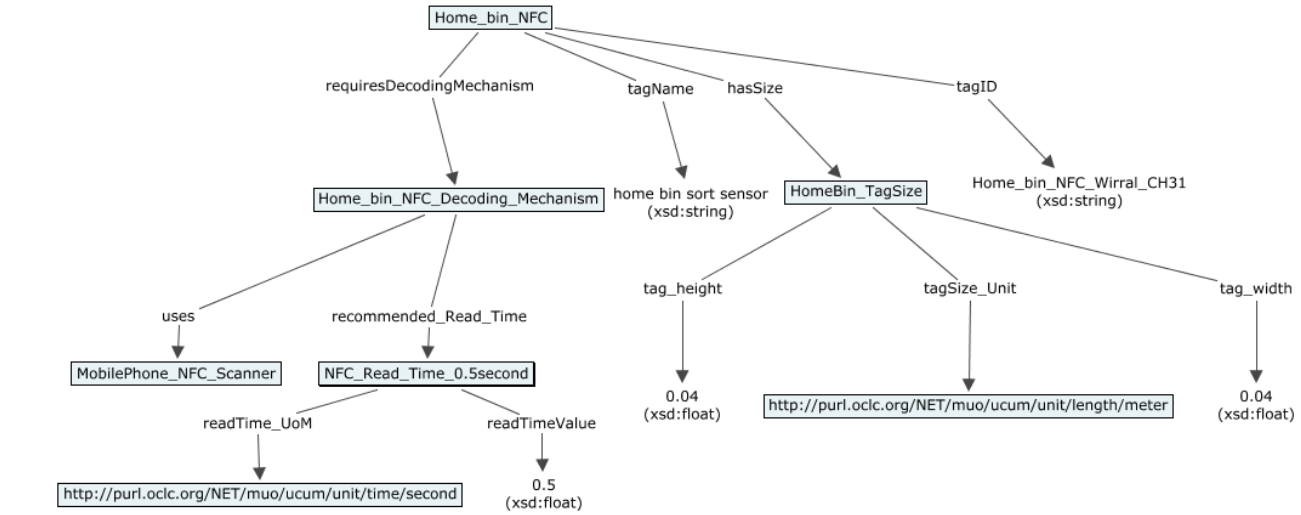
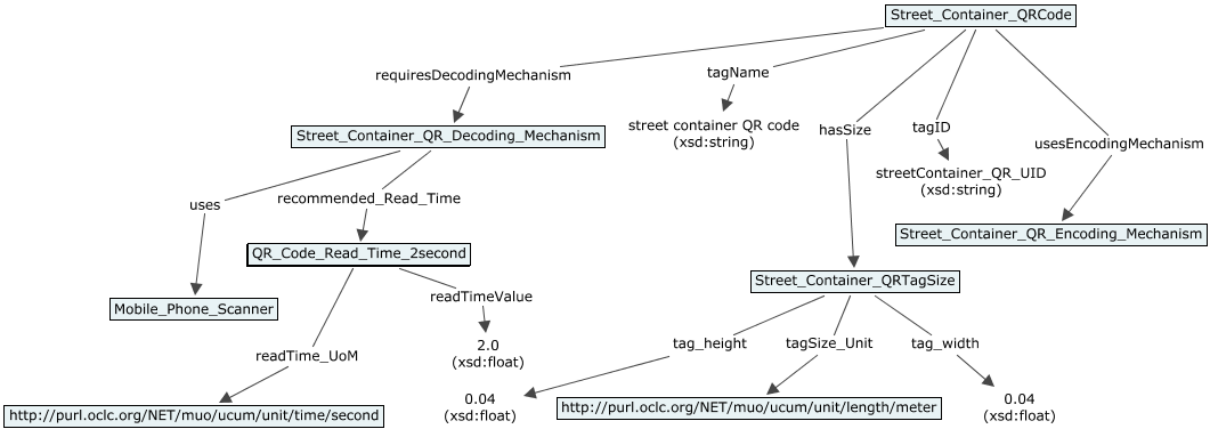
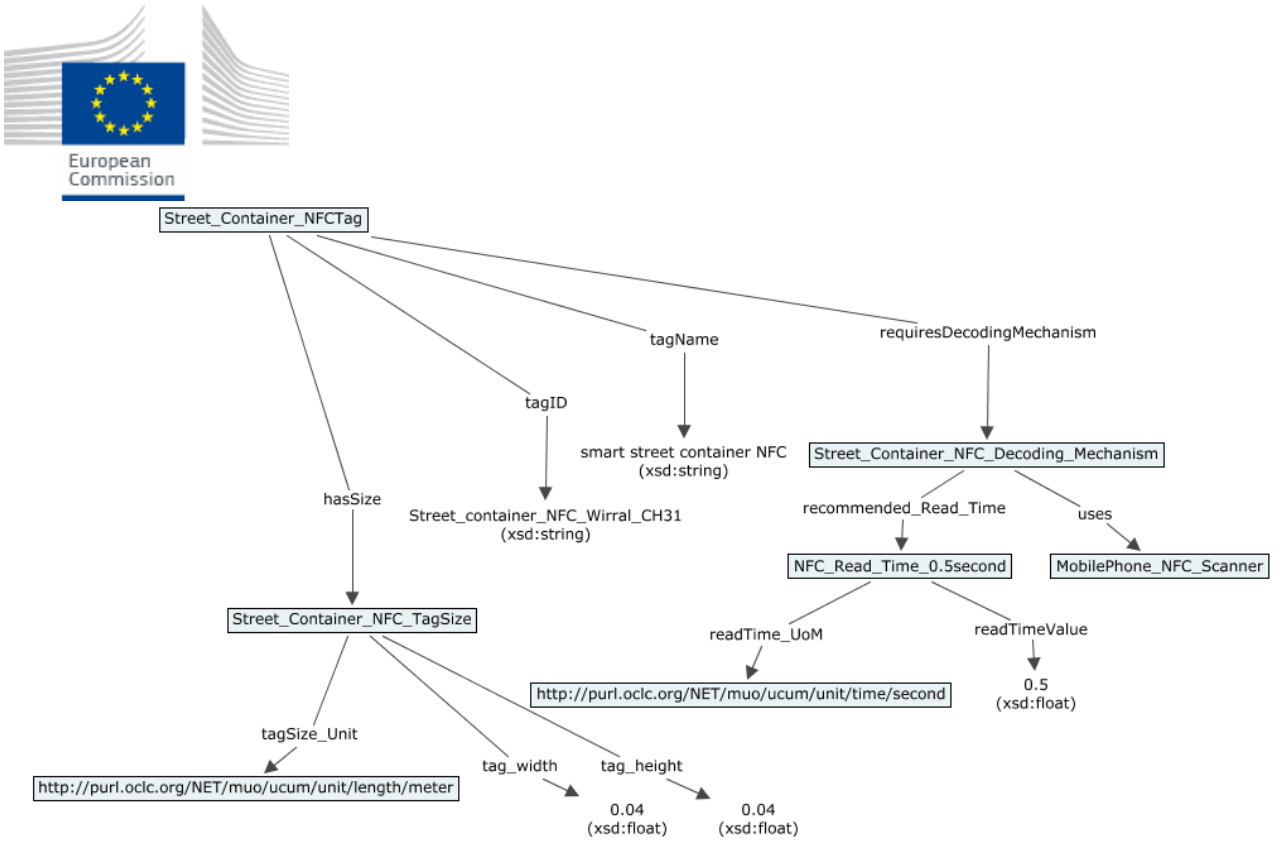


Figure 12: Semantic model instance of Home Bin NFC tag (Lifecycle management UC)

For the non-combined smart street container label case, the semantic model is instantiated with individuals of the corresponding *QR_Code* and *NFCTag* classes, with each having its unique ID and properties, as shown in Fig. 13 (a) and (b), respectively.



(a)



(b)

Figure 13: Smart Street container (a) QR code and (b) NFC tag instances

4.2.2 Time-temperature indicator: coding

Encoding the TTI indicator is done by applying a sensor area close to existing QR code, during printing of the QR code. The printed sensor area is invisible in temperatures below -10 C° and above -10 C° over 30 minutes a red area becomes visible.

4.2.3 NFC sensor: coding

Encoding the NFC sensor is done by applying a NFC sensor close to an existing QR code, after printing the QR code.

4.3 Brand protection

The general aspect of the use case brand protection is to create a platform for originality proof of consumer goods at each part of the logistic chain and at the customer side for a broad variety of products such as high-performance brand (perfume, handbags, sport shoes) or medicine (vitamins, pain reliefs, vaccines). The originality proof tool consists of both a SmartTag as well as the corresponding encoding / decoding mechanism.

SmartTags consist of textures that stay in their original shape during the readout period and textures, that significantly change the outer appearance in a way that in the activated stage additional information is released. The SmartTag design changes within a certain lot size (e.g. per batch, per day); therefore, the uniqueness for a limited amount of products is ensured – the encoded information is stored at a corresponding cloud. The frequent change in the tag design which has to be at least on a daily basis justifies the usage of digital printing technology.

4.3.1 Semantic Model for SmartTags used in this use case

The Brand protection use case is based on photochromic inks reacting to UV sources or thermochromic inks reacting to temperature in a range between 25-40 °C. The QR code is printed with regular inks and one of the functional inks (parts of code). The functional ink changes from colourless to colored when exposed to either UV light or temperature in a range between 25-40 °C. The colour changes are reversible. The corresponding SmartTag model instance for a functional ink that is reversibly activated by UV light is shown in Fig. 14.

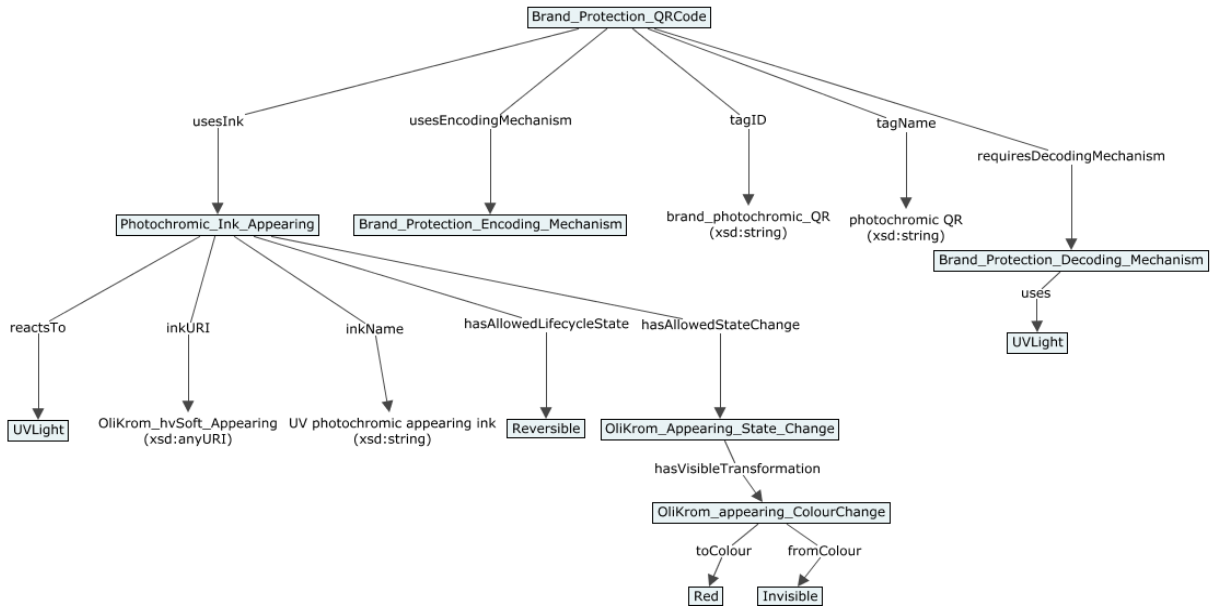


Figure 14: Semantic model of Brand Protection use case

4.3.2 Encoding of sensor

The Brand protection use case is based on photochromic inks reacting to UV light or thermochromic inks reacting to temperature in the range between 25-40 °C. The QR code is printed with regular inks and a photochromic or thermochromic ink (parts of code). The functional ink changes from invisible to colour (appearing) when exposed to camera flash. The colour changes are reversible. An example instantiation of the SmartTag semantic model is shown in Fig. 15, for the particular case where a UV-reactive photochromic ink has been used for the printed QR code tag.

Printing workflow for Brand protection use case is presented in Figure 15. This is quite a similar process to Digital product use case except the whole label is printed with digital printing (inkjet). The sensor is based either on a photochromic ink or a thermochromic ink.

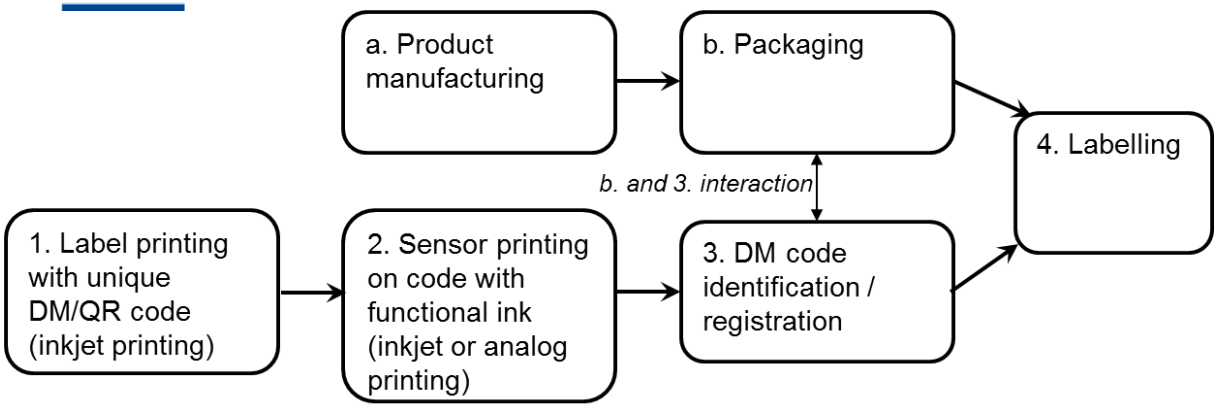


Figure 15: Printing workflow for Brand protection use case.

4.4 Dynamic pricing

In this use case sensor is thermochromic sensor with special requirement for the temperature that should rise above (+7 ◦C) more than 30 minutes. Encoding and decoding process for this sensor is straightforward; sensor is embedded.

4.4.1 Semantic Model for SmartTags used in this use case

The dynamic pricing UC features a QR code with an irreversible thermochromic ink that features a colour change from black to red after being exposed to +7°C for 30 minutes. In this case, all three possible conditions of a functional ink's state change are manifest, namely, the visible colour transformation, the time lapse and the threshold. The resulting semantic model instance of the QR code is shown in Fig. 16.

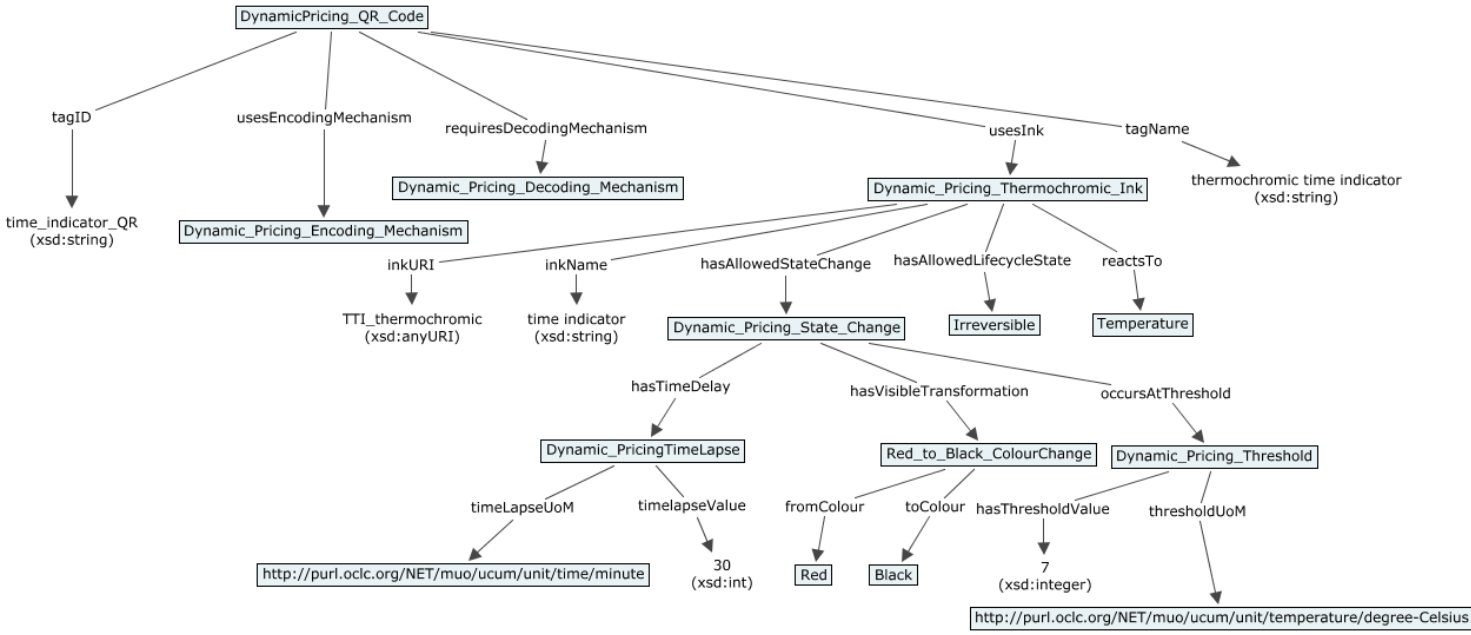


Figure 16: Semantic model of Dynamic Pricing use case

4.4.2 Encoding of sensor

The encoding for SmartTag is based on a QR code extended with a branded logo image printed below the QR code and sensor encoded as a part of a logo (Figure 17). The red part representing the senor is printed with functional ink.

Figure 17: QR code with image and sensor

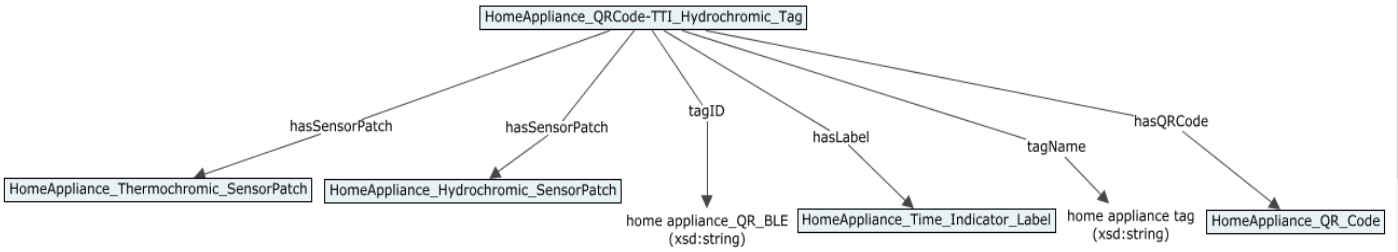
The SmartTag encoded using this approach represents 25-digit alphanumeric value (product code, product line). The functional ink presented in red color will disappear when temperature rises above the predefined threshold. Printing of the functional ink is done separately from printing of the QR code and is more effective from the FC-reader perspective if the functional ink part is printed outside of the QR code. The printing is done during two-phases: first, QR code is printed (potentially with logo) and then as a second step sensor is “embedding” in predefined position.

4.5 Home Services

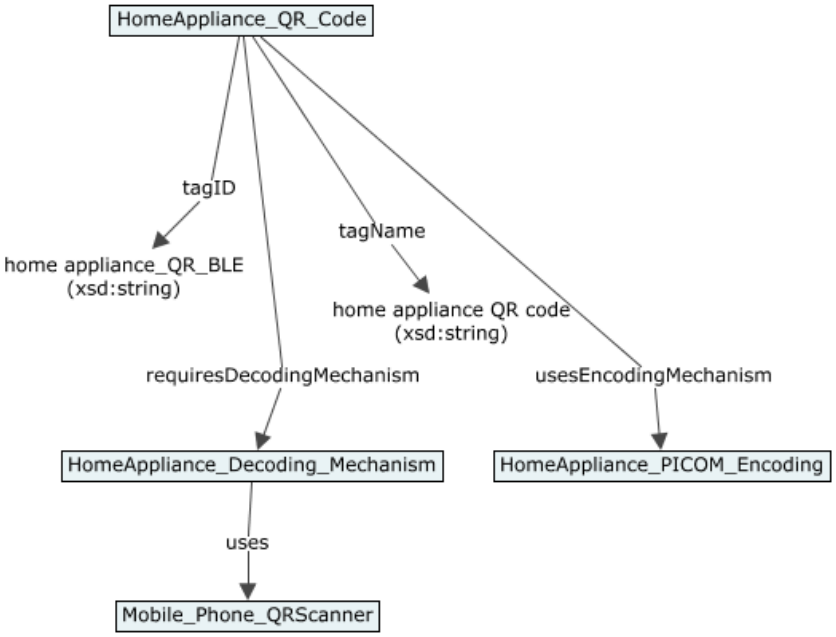
Home services use case is based on three functional inks or indicators: 1) an irreversible thermochromic ink that changes from clear to visible when at temperature >+50 ^{o}C, 2) a time elapse indicator that contains a progressing red liquid front (the front will become visible on the label after 3 months) and 3) hydrochromic ink reacting to the presence of water. It is worth noting that based on the use case evolution and features to be considered, some additional SmartTag types for this use case may be introduced.

4.5.1 Semantic model for SmartTags used in this use case

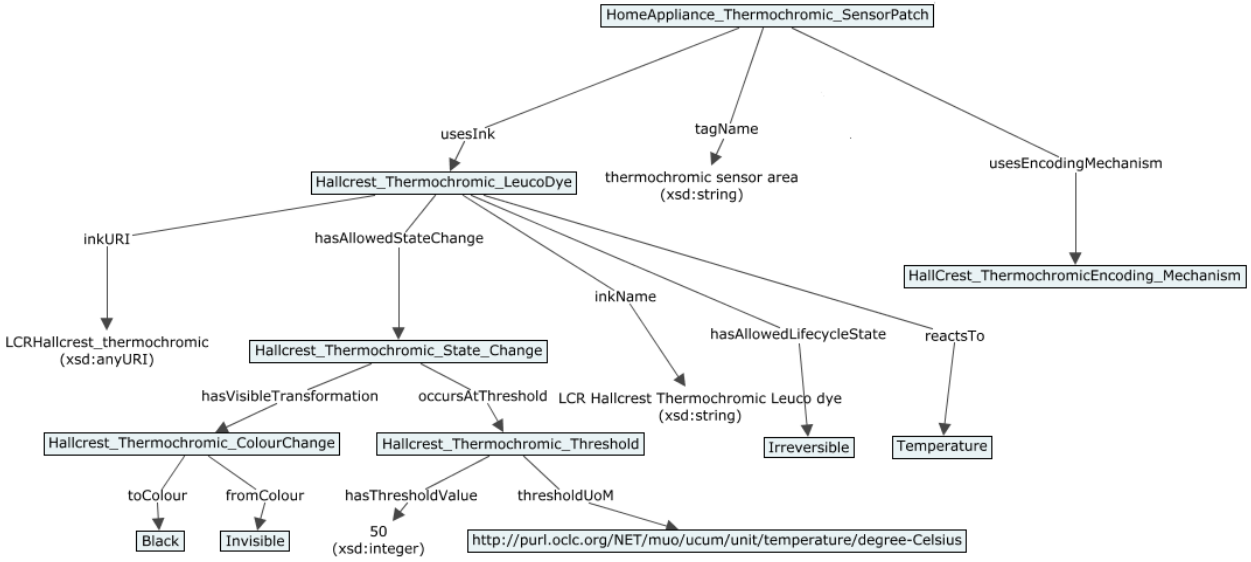
In this use case a SmartTag label consisting of a customized QR code also features three perforated areas below the QR code, that represent respectively, sensor areas corresponding to thermochromic, hydrochromic and time indicator labels. The thermochromic and hydrochromic sensor areas are printed using the respective environment-reactive functional inks and the time indicator area is a printed label that measures time up to 3 months. Fig. 18 (a) shows the top-level view of the resulting SmartTag instance, with subsequent diagrams depicting the different parts of the SmartTag.


HomeAppliance_QRCode-TTI_Hydrochromic_Tag
hasSensorPatch
hasSensorPatch
tagID
hasLabel
tagName
hasQRCode
HomeAppliance_Thermochromic_SensorPatch
HomeAppliance_Hydrochromic_SensorPatch
home appliance_QR_BLE
(xsd:string)
HomeAppliance_Time_Indicator_Label
home appliance tag
(xsd:string)
HomeAppliance_QR_Code

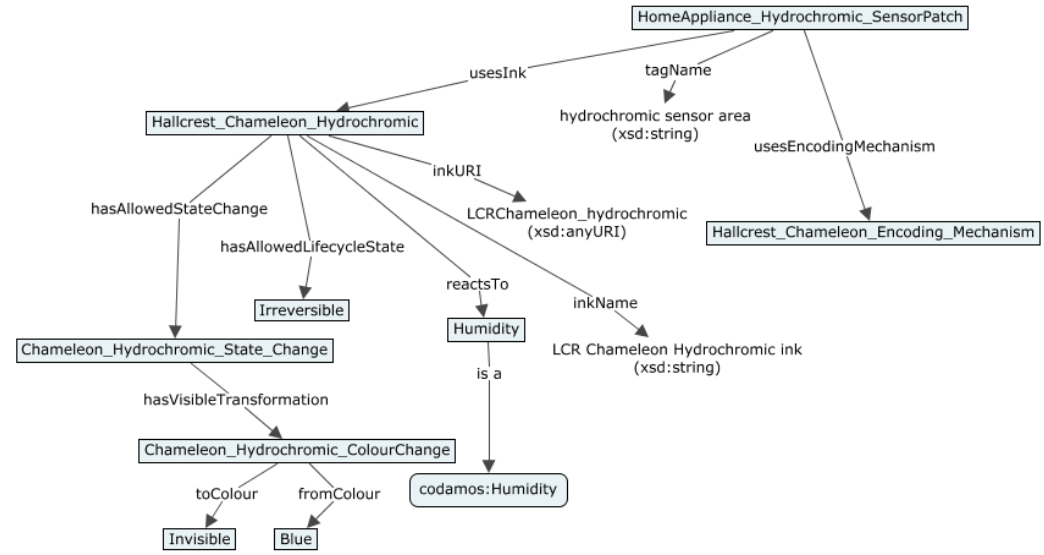
(a)


HomeAppliance_QR_Code
tagID
home appliance_QR_BLE
(xsd:string)
tagName
home appliance QR code
(xsd:string)
requiresDecodingMechanism
usesEncodingMechanism
HomeAppliance_Decoding_Mechanism
HomeAppliance_PICOM_Encoding
uses
Mobile_Phone_QRScanner

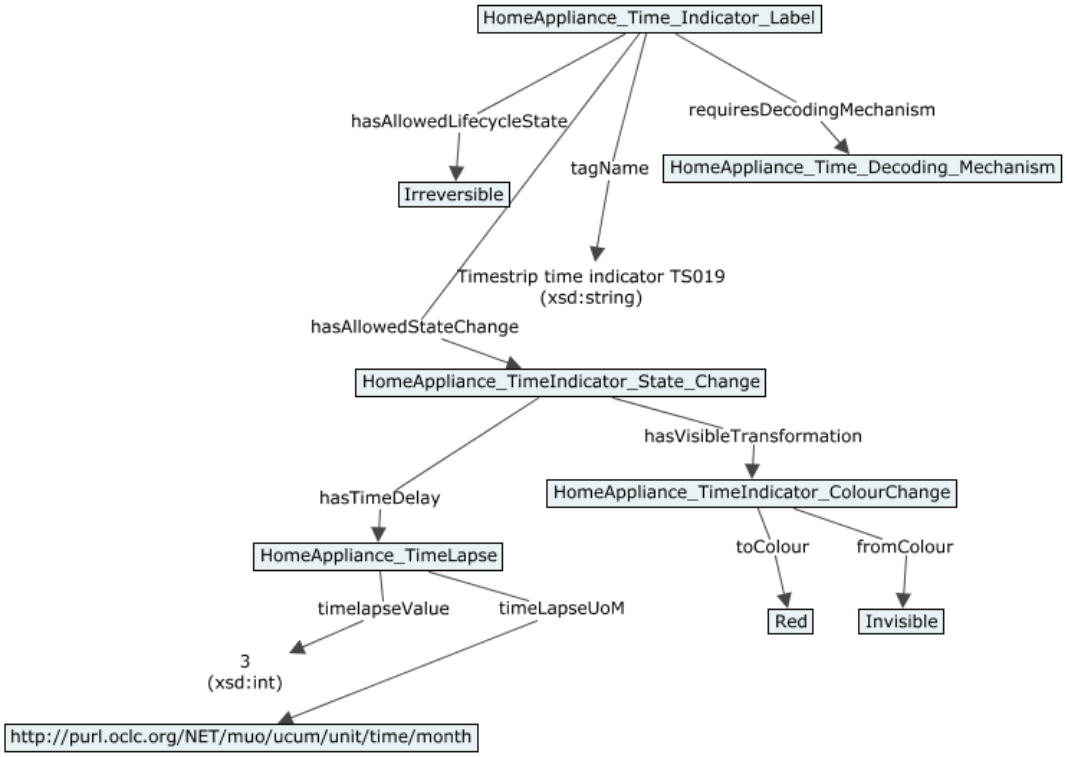
(b)


HomeAppliance_Thermochromic_SensorPatch
usesInk
tagName
usesEncodingMechanism
Hallcrest_Thermochromic_LeucoDye
thermochromic sensor area
(xsd:string)
inkURI
hasAllowedStateChange
HallCrest_ThermochromicEncoding_Mechanism
inkName
hasAllowedLifecycleState
reactsTo
LCRHallcrest_thermochromic
(xsd:anyURI)
Hallcrest_Thermochromic_State_Change
hasVisibleTransformation
occursAtThreshold
LCR Hallcrest Thermochromic Leuco dye
(xsd:string)
Hallcrest_Thermochromic_ColourChange
Hallcrest_Thermochromic_Threshold
Irreversible
Temperature
toColour
fromColour
hasThresholdValue
thresholdUoM
Black
Invisible
50
(xsd:integer)
http://purl.oclc.org/NET/muo/ucum/unit/temperature/degree-Celsius

(c)



(d)



(e)

Figure 18: Semantic model of Home Services use case, showing (a) main classes, (b) the QR code part, (c) the thermochromic sensor area, (d) the hydrochromic sensor area and e) the time indicator label.

4.5.2 Encoding of sensor

The encoding of sensors in SmartTag is done with perforation of label that will contain sensors and the QR code. The placeholder for sensors is used to embed the 1-3 sensors (Figure 19) as a part of physical encoding process. Label with sensors and QR code is then attached to the device (home appliance).

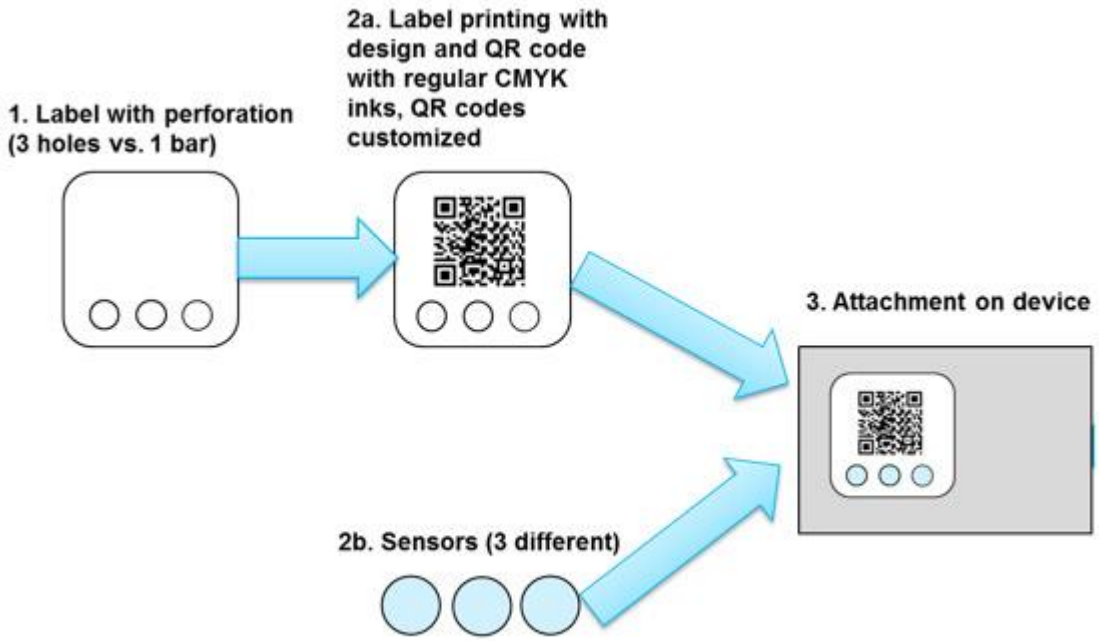


Figure 19: Sensor label encoding in the Home Services use case

Section 5 - Conclusions

This deliverable presented the initial version of the enablers for FunCodes. These enablers address the phase of SmartTag creation, involving secure encoding of sensor information and identity provided through the Identity Management enabler as well as printing considerations. This deliverable also presented needed updates to the SmartTags semantic model, introduced to account for further developments in the project. The semantic model was also validated through use case specific instantiations that showed its ability to represent SmartTags created also as combinations of technologies. It is worth noting that the semantic model is generic enough to account for further combinations, not currently used in the project's considered use cases.

The next iteration of this deliverable (D2.3 at M24) will present implementation details for the enablers presented here in first specification detail, as well as the results of their validation in the context of the field trials based on their integration with the overall system and the interactions with the counterpart enablers in WP3.

Section 6 - References

[1] TagItSmart D5.2 deliverable - Operational trial on core platform integration, March 2017.
[2] Chuang, Jun-Chou, Yu-Chen Hu, and Hsien-Ju Ko. “A novel secret sharing technique using QR code.” International Journal of Image Processing (IJIP) 4.5 (2010): 468-475.
[3] Lin, Pei-Yu, et al. “Secret hiding mechanism using QR bar code.” Signal-Image Technology & Internet-Based Systems (SITIS), 2013 International Conference on. IEEE, 2013.
[4] Lin, Pei-Yu. “Distributed Secret Sharing Approach with Cheater Prevention Based on QR Code.” IEEE Transactions on Industrial Informatics 12.1 (2016): 384-392.
[5] Shamir, Adi. "Identity-based cryptosystems and signature schemes." Workshop on the Theory and Application of Cryptographic Techniques. Springer Berlin Heidelberg, 1984.
[6] Sahai, Amit, and Brent Waters. "Fuzzy identity-based encryption." Annual International Conference on the Theory and Applications of Cryptographic Techniques. Springer Berlin Heidelberg, 2005.