

Universal Messaging Standards for the IoT from a Lifecycle Management Perspective

Kary Främling, Sylvain Kubler, Andrea Buda

Abstract—According to our original vision of the Internet of Things, it should be possible to create ad hoc and loosely coupled information flows between any kinds of products, devices, computers, users and information systems in general when and as needed. However, this is still challenging to achieve in practice due to the lack of sufficiently generic and standardized interfaces for creating the needed information flows between all devices and systems that the IoT is composed of. The paper presents necessary requirements for such interfaces, as well as proposed interface standards that fulfill those requirements. The paper describes the design principles and provides a high-level description of the proposed standards, followed by real-life implementations that illustrate why such standards are needed and how they are applied.

Index Terms—Internet of Things; Cyber Physical System; Quantum Lifecycle Management; Product lifecycle management; Messaging standards; Intelligent products.

I. INTRODUCTION

NEW challenges and opportunities arise with concepts such as the Internet of Things (IoT) or the so-called Cyber Physical Systems (CPS) [1], [2]. Through these concepts, objects of the real world are linked with the virtual world, thus enabling connectivity anywhere, anytime and for anything. It refers to a world where physical objects and beings, as well as virtual data and environments, may interact with each other when and as needed [3]. In short, connections are not just people to people or people to computers, but people to things as well as things to things. Many applications in various sectors exist: medical [4], automotive [5], home automation [6], manufacturing [7] and much more.

In practice, for many years the term IoT became a synonym for Radio Frequency Identification (RFID) technology and supply chain management related applications [8]. This led to the emergence of a whole family of standards based on the Electronic Product Code (EPC) [9] and its use for tracking stock keeping units of different kinds in the supply chain. A quite different interpretation of the IoT was presented e.g. in [10] as well as in [11], where the IoT was used in the sense of a generic information system for accessing and synchronizing any kind of product-related information, mainly over the Internet. The application scope was in this

case closed-loop Product Lifecycle Management (PLM) [12], [13] and similar domains. In other terms, the focus is given to the entire Product Life Cycle (PLC) from beginning of life (BoL) including design, production, and supply chain tracking and tracing, through Middle of Life (MoL) including use and maintenance, up to End of Life (EoL) including recycling and disposal of the product [14].

The wide scope covered by those applications gave way to complex heterogeneous enterprise system combinations to meet the organization and customer needs. However, as the system complexity increases, its infrastructure and maintenance costs also increase. Organizations are therefore looking for ways to minimize Information and Communications Technology (ICT) costs, while offering better products and services to customers [15]. When considering IoT implementation, finding the right balance is a challenging task for organizations since their ICT infrastructure has to make it possible to easily set up information flows between any kinds of products, devices, computers, users and information systems in general. Although the potential of IoT, CPS or closed-loop PLM are widely recognized and have started to address this challenge, there are still fundamental questions and issues that need to be addressed. Handling the many changes of organization infrastructures and needs throughout the PLC have yet to be designed; developing new strategies for context-aware services, i.e. services able to self-adapt autonomously depending on current conditions, is becoming a glaring demand [16], [17]. These challenges have to be addressed in order to provide IoT architectures that are sufficiently flexible to be used in any phase and area of a PLC, regardless of the product context and environment. The notion of *flexibility* is the watchword to design the IoT of tomorrow and to enhance product lifecycle management [18].

The paper has two main objectives: 1) present a set of necessary requirements that need to be satisfied by any IoT messaging standard, which have been identified based on over ten different real-life systems implemented for closed-loop PLM in different domains, and 2) present the Quantum Lifecycle Management (QLM) standards, which have been developed for fulfilling those requirements.

Section II provides the IoT background from a PLM perspective to understand the general interests of introducing new messaging standards. Section III describes the design principles and provides a high-level description of QLM messaging. Section IV provides real-life implementations that show the importance and flexibility of these standards. Section V provides an overview of related standards, followed by conclusions.

K. Främling is with the School of Science and Technology, Aalto University, P.O. Box 15500, FI-00076, Finland (e-mail: kary.framling@aalto.fi)

S. Kubler is with the School of Science and Technology, Aalto University, P.O. Box 15500, FI-00076, Finland (e-mail: sylvain.kubler@aalto.fi)

A. Buda is with the School of Science and Technology, Aalto University, P.O. Box 15500, FI-00076, Finland (e-mail: andrea.buda@aalto.fi)

Copyright (c) 2012 IEEE. Personal use of this material is permitted. However, permission to use this material for any other purposes must be obtained from the IEEE by sending a request to pubs-permissions@ieee.org.”

II. IOT BACKGROUND FROM A PLM PERSPECTIVE

The phrase product lifecycle management (PLM), as well as product lifecycle information management (PLIM), is commonly understood to be a strategic approach that incorporates the management of data associated with products of a particular type, and perhaps the versions and variants of that product type, as well as the business processes that surround it [19], [20]. These product definition data are generated when the product is first conceived, and it then continues to evolve with the addition of detailed specifications, user manuals, computer aided design (CAD) drawings, manufacturing instructions, service manuals, disposal and recycling instructions and so forth. For such traditional PLIM, the product information generation process seems to end after production. When the product enters actual use, PLIM mainly signifies providing access to the existing information but hardly any new information is generated about the products.

This mainly reflects the point of view of the manufacturing industry that tends to see PLIM mainly as a distributed knowledge management task of the “extended enterprise” [21] that created the product. With this view of PLIM, there has been only slight interest in how the customer uses each individual product, or in how that product has behaved. Concepts such as “Product Agents” [22], “product-centric” PLIM [23] and “Intelligent Products” [24] have been proposed as solutions for enabling such item- or instance-enabled PLIM.

Such concepts were the cornerstones of the product instance-enabled PLIM solutions developed in the PROMISE EU FP6 project¹. QLM standards emerged out of the PROMISE project, where real-life industrial applications required the collection and management of product instance-level information for many domains involving heavy and personal vehicles, household equipment, phone switches, *etc.* Information such as sensor readings, alarms, assembly, disassembly, shipping event, and other information related to the entire PLC needed to be exchanged between products and systems of different organizations. Based on the needs of those real-life applications, the requirements listed in TABLE I were identified. As no existing standards could be identified that would fulfill those requirements without extensive modification or extensions, PROMISE partners started the specification of new messaging interfaces. Those specifications have since then been further developed by the QLM workgroup of The Open Group² with the explicit goal to publish them as standards for the IoT. QLM messaging specifications consist of two standards proposals [25]: the QLM Messaging Interface (QLM-MI) that defines a set of possible interactions between entities, while the QLM Data Format (QLM-DF) specifies a generic structure for the IoT payload information included in a QLM message.

III. QLM STANDARDS

In the QLM world, communication between the participants, e.g. products and backend systems, is done by passing messages between nodes using QLM-MI. The QLM “cloud” in

TABLE I
FUNCTIONAL REQUIREMENTS FOR IOT MESSAGING

-
- Possible to implement for any kind of instances as independently of the application domain as possible
 - Possible to implement for any kind of information systems, including embedded and mobile systems
 - Support for “synchronous” messaging such as immediate read and write operations, including “client-poll” subscriptions
 - Not restricted to one communication protocol only, it must be possible to send messages using protocols such as plain Hypertext Transfer Protocol (HTTP), Simple Object Access Protocol (SOAP), Simple Mail Transfer Protocol (SMTP), as file copies, *etc.*
 - Possibility to create ad hoc, loosely-coupled, time-limited information flows “on the fly”
 - Peer-to-peer communication possibility for all devices, i.e. client and server functionality can be implemented for any device, depending on available processing power, network connectivity, *etc.*
 - Handling mobility and intermittent network connectivity, i.e. support for asynchronous messaging capabilities that imply for instance message persistence, time-to-live, *etc.*
 - Context-dependent discovery of instances, instance-related services and meta-data about them
 - Support for context- and domain-specific ontologies
 - Queries by regular expressions for retrieving information about more than one instance and more than one kind of information
 - Historical queries, i.e. retrieving values between two points in time
-

Fig. 1(a) is intentionally drawn in the same way as the Web cloud. Where the Web uses the HTTP protocol for transmitting HTML-coded information mainly intended for human users, QLM-MI is used for transmitting QLM-DF represented IoT information mainly for processing by information systems. In the same way as HTTP can be used for transporting payloads also in other formats than HTML (such as XML and QLM-MI messages), QLM-MI can be used for transporting payloads also in other formats than QLM-DF. QLM-DF fulfills the same role in the IoT as HTML does for the Internet, meaning that QLM-DF is a generic content description model for things in the IoT.

QLM specifications are written using XML schema due to its flexibility for describing complex data structures. However, QLM messages can also be represented using JavaScript Object Notation (JSON) and other formats that can be translated directly to and from XML. Information encoded using QLM-DF can be used as payload also when using plain TCP/IP, HTTP or similar protocols. Indeed, QLM-MI and QLM-DF are independent entities that reside in the Application layer of the OSI model, as illustrated in Fig. 1(b), where QLM-MI is specified at the Communication level and QLM-DF is specified at the Format level. Therefore, both standards can be used independently of each other.

A. QLM Data Format

QLM Data Format (QLM-DF) is defined as a simple ontology, specified using XML Schema, that is generic enough for representing “any” object and information that is needed for information exchange in the IoT. It is intentionally defined in a similar way as data structures in object-oriented programming.

It is structured as a hierarchy with an “Objects” element as its top element. The Objects element can contain any number of “Object” sub-elements. Fig. 2 gives insight into both the generic hierarchy/object tree and an example of

¹<http://promise-innovation.com>

²<http://www.opengroup.org/qlm/>

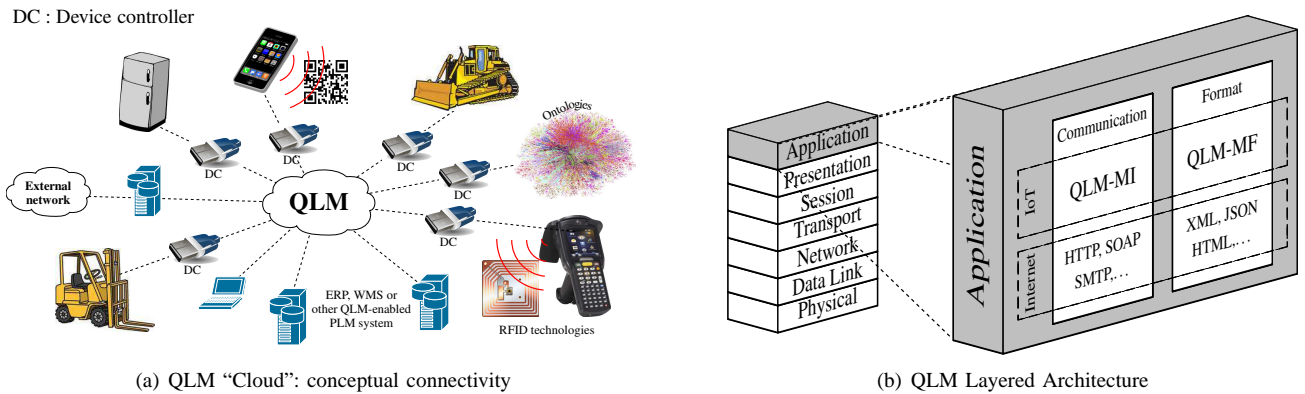


Fig. 1. QLM messaging standards: QLM-MI & QLM-DF

a QLM message whose structure relies on that object tree. In this example, a unique object of type Refrigerator (see row 5 of the XML message) is considered. Object elements can have any number of properties, referred to as InfoItem(s), as well as Object sub-elements. In our example, the object Refrigerator has two InfoItems named Door Led Status and Fridge thermostat (see row 7 and 10 respectively). The resulting Object tree can contain any number of levels. The most important attribute of an Object is “type”, which specifies what kind of object it is. An optional attribute called “udef” may be used for specifying the object class using the Universal Data Element Framework (UDEF) taxonomy³, which is a standard for indexing enterprise information. Every Object has a compulsory sub-element called “id” that identifies the Object (see row 6 in Fig. 2). The id should preferably be globally unique as described in [26] or at least unique for the specific application, domain or network of the involved organizations. InfoItems can contain the following optional sub-elements:

- *Description*: text intended mainly for human user interfaces that explains what the InfoItem is;
- *MetaData*: sub-element that provides meta-data information about the InfoItem, such as value type, units and other similar information;
- *Value*: arbitrary number of values for the InfoItem, possibly with timestamps.

Even though it is possible to include all these sub-elements in the InfoItem, only one of them is usually included. MetaData is typically requested for only once when encountering a previously unknown InfoItem. The MetaData element can contain an arbitrary number of elements. MetaData elements are also of InfoItem type because they are syntactically similar to Object InfoItems, even though MetaData InfoItems are conceptually different from Object InfoItems. The “description” element could also be considered as MetaData. However, it has been left as a separate element mainly due to experiences that have shown the utility of including a simple-to-use “free-form” text element for user interface and debugging purposes. Value element(s) contain actual data.

The high flexibility of the Objects tree makes it possible to respect a precise structure or data model defined in an

application or by a standard. QLM-DF defines an extension mechanism that makes it possible to use class inheritance in similar ways as in object-oriented programming. This extension mechanism enables the creation of domain-specific extensions of QLM-DF, while preserving a basic compatibility between different domain extensions. For the moment, the QLM workgroup has created one such extension, called the *Physical Product Extension*, which provides specifications for representing PLIM information [27].

In object-oriented programming, objects are aware of each other both by object containment hierarchies and by reference or pointers. In QLM-DF, such object references are made by using the Object id element. However, in the IoT the id does not refer to a specific memory location but to an IoT object whose information may even be spread over several information systems and organizations. Different methods and systems have been proposed for the discovery of such distributed information, e.g. in [26]. The simplest mechanism is to include a URL in the identifier itself as proposed by Huvio et al. [28], and then to retrieve the information by object linking as proposed by Främling et al. [29]. However, new methods are still being developed for solving this issue, which are out of the scope of this paper.

B. QLM messaging interface

A defining characteristic of the QLM Messaging Interface (QLM-MI) is that QLM nodes may act both as a “server” and as a “client”, and therefore communicate directly with each other or with back-end servers in a peer-to-peer manner. Typical examples of exchanged data are sensor readings, lifecycle events, requests for historical data, notifications, etc. The main properties of QLM-MI are listed in TABLE II.

One of the fundamental properties of QLM-MI is that QLM messages are “protocol agnostic” (see property 1), so they can be exchanged using HTTP, SOAP, SMTP, File Transfer Protocol (FTP) or similar protocols. The most appropriate protocol to use depends on the application as well as on the security mechanisms of the protocols. This non-dependency on specific communication protocols makes QLM different from many (or most) other potential IoT messaging standards.

The subscription mechanism and its variants (see property 2: Read) is a cornerstone of QLM-MI. The conceptual framework

³www.udef.com

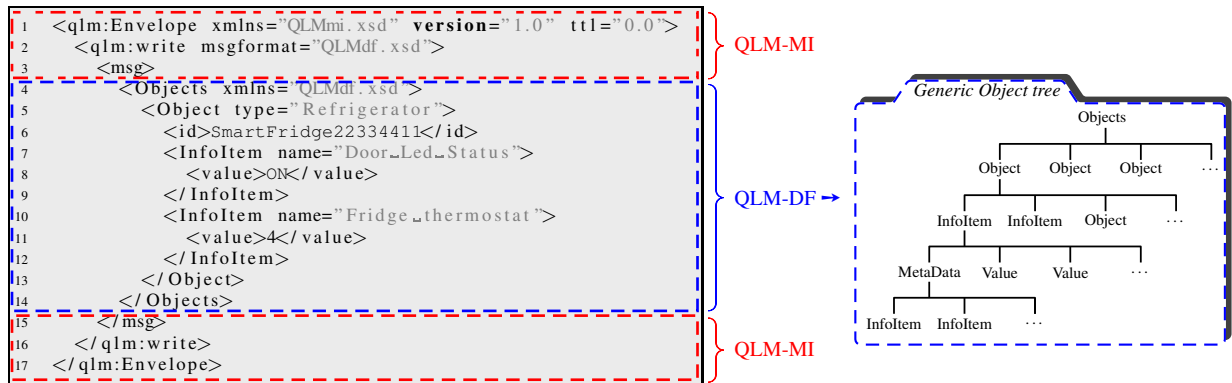


Fig. 2. QLM Data Format: generic object tree and example of a QLM message relying on that tree

used here is the Observer Design Pattern presented by [30] and applied according to [29], which signifies that a QLM node can add itself as an observer of events that occur at another QLM node. In this sense, QLM differs from e.g. Java Message Service (JMS), which is based on the Publish/Subscribe (Pub/Sub) model. For many applications, the Observer and the Pub/Sub models can be used in quite similar ways. However, the Pub/Sub model usually assumes the use of a high-availability server, which the Observer pattern does not [31]. This is why the Observer model is more suitable for IoT applications where products might communicate with each other directly. Two types of subscriptions can be performed with QLM-MI, which both rely on the QLM Read operation:

- 1) *Deferred retrieval of information with callback address*: the data is sent to the subscriber QLM node using a QLM response at the requested interval. The interval parameter is specified by the subscriber when requesting for the subscription. Two types of intervals can be defined: *i*) Interval-based (interval > 0), and *ii*) Event-based (interval = -1);
- 2) *Deferred retrieval of information without callback address*: the data is memorized on the subscribed QLM node as long as the subscription is valid⁴. The memorized information can be retrieved (i.e., polled) by issuing a new QLM read query containing the ID of the subscription.

QLM messages are “self-contained” in the sense that all the necessary information to enable the recipient to handle the message is contained in the message itself (e.g., the actions to be performed: read, write, subscription...); the TTL; the callback address; etc. The QLM message given as example in Fig. 2 shows the interface and such information (row 1 provides the TTL value and row 2 indicates that this message has to be processed as a “write” request).

QLM-MI is designed to be RESTful [32] to the extent that it has been possible and practical. This signifies that QLM messages will normally contain enough information to enable QLM nodes to avoid maintaining state information (such as

⁴In case of a subscription request, the time-to-live (TTL) parameter defined in QLM-MI is used to indicate the period of validity of the subscription. A TTL of -1 indicates that the subscription is “forever” (i.e., it is valid as long as the subscriber does not cancel it or the subscribed node removes it for some reason).

sessions or similar) as far as possible. Subscriptions are the only exception because they have to be memorized as long as they are valid. However, it is hard to imagine how a stateless subscription could be implemented. RESTful interfaces are usually expected to use only the basic HTTP operations GET, POST, PUT, and DELETE. The GET operation is usually used for directly retrieving information by entering a corresponding URL. In practice, such a request corresponds to the simplest possible QLM Read request. All other QLM messages are expected to be exchanged using the POST method when sending them over HTTP. We do not intend to use PUT and DELETE methods because that would create an explicit link between QLM-MI and the HTTP protocol, which would go against the functional requirements set out for the messaging interface (TABLE I). Indeed, we consider that it is not a good praxis to use all the GET, POST, PUT, and DELETE operations for implementing web-based services because it makes the services HTTP-specific.

IV. IMPLEMENTATIONS

Two implemented applications using QLM messaging, respectively defined in BoL and between MoL-BoL, are presented in sections IV-A and IV-B.

A. Production line of car chassis

This scenario is a real case study from the LinkedDesign EU FP7 project⁵, in which different actors work on a production line of car chassis as illustrated in Fig. 3: chassis parts are first moved from the oven to a press machine, and then to other operations. This process segment involves two robots to transfer the chassis part from machine to machine. The actors involved in the manufacturing plan expressed, on the one hand, the need to check each chassis part throughout the hot stamping process and, on the other hand, the need to define communication strategies adapted to their own needs. Accordingly, scanners are added between each operation for the verification procedure (see Fig. 3), and QLM messaging is adopted to provide the types of interfaces required by each actor. QLM nodes were installed at the relevant physical nodes of the production line, namely: the two scanners; a server in

⁵<http://www.linkeddesign.eu>

TABLE II
MAIN PROPERTIES OF THE QLM MESSAGING INTERFACE (QLM-MI)

Property	Description
1-Protocol agnostic	QLM supports multiple underlying protocols, making it possible to transport the message using most “lower-level” protocols such as HTTP, SOAP, SMTP, FTP, etc.. It might also be possible to transport this message using files on USB sticks or other memory devices
2-Three possible operations	1) Write: used for sending information updates to QLM nodes 2) Read: used for immediate retrieval of information and for placing subscriptions for deferred retrieval of information from a node 3) Cancel: used to cancel subscriptions before they expire
3-Time-to-live	If the message has not been delivered to the “next” node before time-to-live (TTL) expires, then the message should be removed and an error message returned to the message originator (if possible)
4-Self-contained message	a QLM message contains all the necessary information to enable the recipient to appropriately handle the message. In a more concrete level, the message contains all the relevant information such as the actions to be performed (read, write, subscription,...), the message validity period (TTL), the mode of communication (asynchronous or synchronous), or still the callback address
5-Multiple payload formats	any QLM message can transport actual information using any text-based format that can be embedded into an XML message. A response may include return elements that correspond to several QLM requests. In that case, it is even possible to use different payload formats in different return elements. However, the return payload would normally be in the same format as the original request payload
6-Real-time communication	QLM allows piggy backing a new request with a response. This is a crucial property both for real-time communications and to enable two-way communications with mobile nodes and/or nodes located behind firewalls
7-Publication and discovery	Publication of new data sources, services and meta-data can be done with QLM write operation. “RESTful” URL-based queries allow the discovery of them, including discovery by search engines
8-Target nodes	Intermediate node(s) are responsible of re-routing messages to the target QLM nodes, or sending back an error message to the requesting QLM node in case of failure

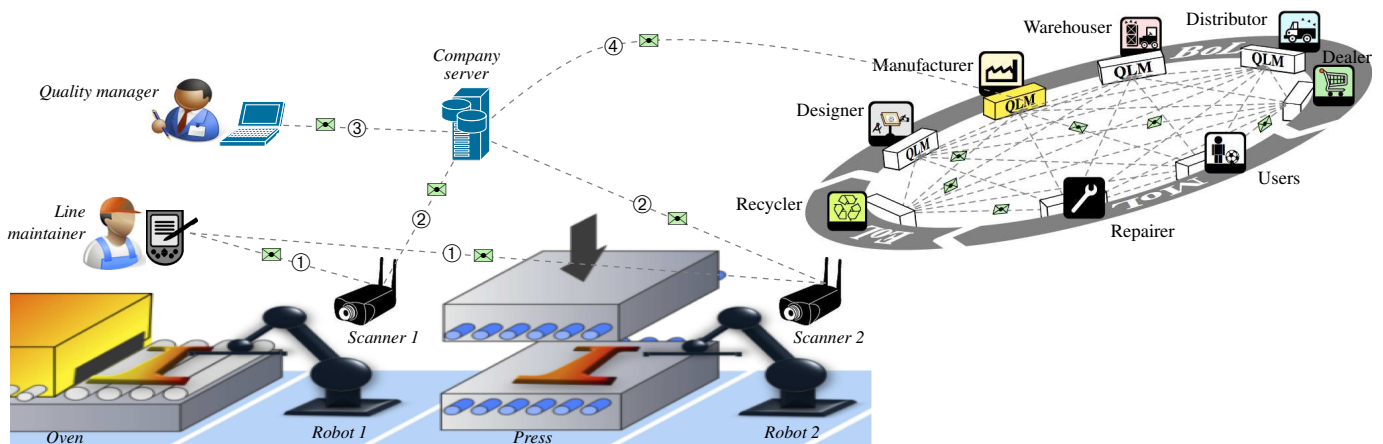


Fig. 3. Hot stamping process implementing QLM messaging for monitoring and control purposes

charge of collecting data generated on the production line; the line maintainer’s personal digital assistant (PDA); the quality manager’s computer.

The line maintainer expressed the need to receive all verification events generated by scanners 1 and 2 to identify in real-time when a problem occurs on a chassis part. Accordingly, the maintainer directly subscribes to these scanners (see communication denoted by ① in Fig. 3). This is made possible by performing a QLM read query by *i*) setting the interval parameter to “-1” to indicate that the subscription is *event-based*, *ii*) by including his own address as callback (i.e., the PDA’s address), and *iii*) by setting the TTL parameter to “-1” that indicates that the validity period of the subscription is “forever”. These parameters are shown in lines 1, 2 and 3 of the subscription request given in Fig. 4. Line 7 indicates the name of the InfoItem to be subscribed to, i.e. *StatusD*.

The sequence diagram in Fig. 5(a) gives insight into the transactions resulting from this subscription. First, a response that contains the ID of the subscription is returned to the PDA. Then, each time a chassis part passes under scanner 1, the subscribed InfoItem value (denoted by *StatusD_x* in Fig. 5(a), *x* being the part reference) is pushed to the PDA through a new

```

1 <qlmEnvelope xmlns="QLM_mi.xsd" version="1.0" ttl="-1">
2   <read msgformat="QLM_df.xsd" interval="-1" callback=
3     "http://207.46.130.1/ServletPDA">
4     <msg>
5       <Objects xmlns="QLM_df.xsd">
6         <Object type="Hot_stamping_machine">
7           <id>HotStamp1223</id>
8           <InfoItem name="StatusD"/>
9         </Object>
10        </Objects>
11      </msg>
12    </read>
13  </qlmEnvelope>

```

Fig. 4. PDA subscribes (forever) to the InfoItem *StatusD* to Scanner 1

QLM response. Based on these events, it is possible to develop scripts, for instance, to raise an alarm if a failure occurs on a part. Such an example is illustrated in Fig. 5(b) where chassis part 3 has a fault⁶. The company server subscribes to scanner events by specifying a callback address, in a similar manner as the line maintainer (see communication denoted by ② in Fig. 3). Accordingly, people internal or external to the organization can access the subscribed data on the server

⁶The development of such scripts is outside the scope of QLM messaging.

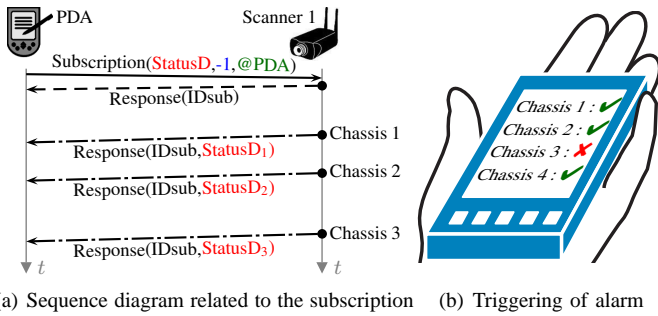


Fig. 5. Subscription with callback address performed by the line maintainer

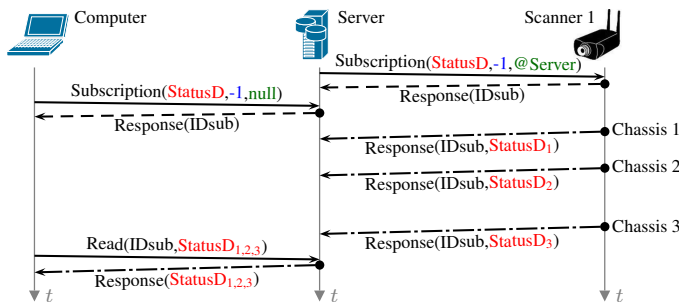


Fig. 6. Subscription without callback address performed by the manager

depending on the security settings.

Unlike the line maintainer, the quality manager is not interested in receiving a continual flow of events from the scanners. His primary function is not to guarantee real-time control, but rather to deal with weekly or monthly evaluations (e.g., to estimate the failure rate over a period of time). Accordingly, the second type of subscription supported by QLM is more appropriate, which consists in retrieving (i.e., polling) one or several historical values on the server by issuing a new QLM read query containing the subscription ID. This corresponds to the communication denoted by ③ in Fig. 3. For this to happen, the quality manager sends a subscription request (via his computer) as illustrated in Fig. 6, by setting the interval parameter to “-1” but, this time, without including a callback address (see the argument “null” in Fig. 6). Then, the server is aware that the events generated by Scanner 1 should be kept locally as long as the subscription is valid. The quality manager can thereafter issue new read queries with the request ID of the subscription for retrieving the collected events as depicted in Fig. 6 (see the Read request with the arguments “IDsub” and the range of stored values denoted by $StatusD_{1,2,3}$ to respectively obtain values of chassis parts 1, 2, and 3).

If other organizations in the PLC support QLM messaging, and if security rules allow them to access particular information from the production line, they could further use the QLM interfaces to subscribe to it, modify it, and so on. This corresponds to the communication denoted by ④ in Fig. 3.

B. Smart house application

The scenario described in this section involves actors from two distinct PLC phases as depicted in Fig. 7(a):

- *in MoL*: a user bought a smart fridge and a TV supporting QLM messaging;
- *in BoL*: the fridge designer agreed with the user to collect specific fridge information over a certain period of the year (June, July, August) using QLM messaging.

Again, the appropriate QLM interfaces regarding each actor have to be set up. The house owner goes on vacation for a period of two weeks and would like to continuously monitor the fridge temperature over that period. Since the owner is not aware of all the features related to his devices (i.e., InfoItems), the RESTful QLM “discovery” mechanism (i.e., property 7 in TABLE II) is used to retrieve the exact InfoItem name(s) to be subscribed to. An example of how this can be achieved using the Unix `wget` utility is shown in Fig. 7(b) with `wget_1`, which returns the set of devices in the house that are reachable using QLM (i.e., `Refrigerator123` and `Television321`). The user then refines his research by retrieving the set of InfoItems related to `Refrigerator123` using `wget_2` in Fig. 7(b), which returns the list of InfoItems that can be accessed for read, write, or subscription on that device. In this case, only one InfoItem is reachable on the fridge, which is named `IndoorTemperature`. Based on this information, the user sends a subscription request to the fridge including:

- *the InfoItem to be subscribed*: `IndoorTemperature`;
- *the callback address*: none, the house QLM node will store the temperature values until the user wants to check them on his mobile device by issuing a new QLM read query including the subscription ID, as described in the first scenario;
- *the interval parameter*: the user does not want to perform an event-based subscription, but would like to obtain the indoor temperature every hour. Accordingly, the interval parameter is set to 3600 (expressed in seconds);
- *the period of the subscription validity*: this corresponds in our scenario to two weeks. This is specified through the TTL parameter.

As with the quality manager in the previous scenario, the statistician is not interested in receiving a continual flow of events from the smart fridge. Rather, he wants to retrieve historical values over a period of time considering a panel of users. These values help to develop learning models and algorithms capable of representing the fridge behavior in diversified environments which, in turn, help to enhance the design of the future generations of fridges. The statistician thus performs a subscription with a specific interval, with a callback address pointing to the QLM node on the fridge designer’s company server, and with TTL equal to 8035200 seconds. This value corresponds to the validity period of the subscription, which is of three months (June, July, August), i.e., $\approx 3 \times 31 \times 24 \times 3600 = 8035200$.

V. COMPARISON WITH EXISTING STANDARDS

When looking for suitable candidates that would fulfill the requirements listed in TABLE I, some entire families of standards were eliminated. One such family of standards are the “low-level” machine-to-machine (M2M) standards, which

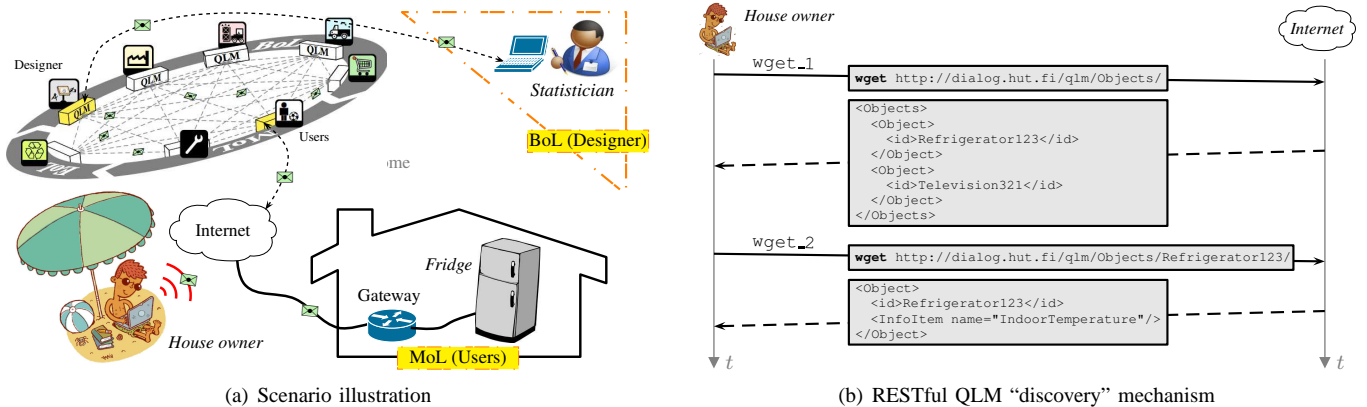


Fig. 7. Application scenario involving the house owner (MoL) and the service provider (BoL)

are usually specified for intranet environments. M2M protocols and standards also tend to be developed for very specific purposes and domains, which makes them unsuitable for use as generic application-level IoT messaging standards. An extensive study of those protocols and standards is therefore out of the scope of this paper.

When considering the PLM aspect of IoT, the requirement for *inter-organizational* communication currently signifies that we are limited to using HTTP or HTTPS as the underlying communication layer. Therefore, comparisons with other potential standards for PLM purposes should be made against standards that can use HTTP or HTTPS. A significant difference between QLM and most such standards is that QLM may also use lower-level transport protocols such as TCP/IP as well as "higher-level" protocols such as SOAP. As pointed out in TABLE II, QLM communications may even use file transfer or USB memories as the underlying protocol. This is an important difference compared to standards such as DPWS (Devices Profile for Web Services), which assumes the usage of SOAP, WSDL and other so-called WS-* standards as underlying protocols. Therefore it is challenging to use DPWS in RESTful services, which is currently the most used approach for developing web applications. With QLM, even radical future changes in the core internet protocols should not make it obsolete. Indeed, QLM has been specified with the objective of re-using existing internet standards as far as possible, while ensuring that QLM is not dependent on any of them.

Despite a continuous survey of existing and upcoming standards, we have not yet identified one that would fulfill all the requirements in TABLE I. A detailed comparison with several relevant standards can be found e.g. in [33] but in practice it is impossible to provide a complete comparison between QLM and all other existing standards or *de facto* standards. Some standards that we have studied in particular are JMS [34] and EPC Information Services [9]. However, JMS was developed for integrating distributed enterprise applications and essentially uses a server-based Pub/Sub model with heavy Java-based server implementations, which makes it difficult to realize M2M communication for low-range computing hardware. EPC Information Services also uses a server-based

philosophy and does not currently support the transmission of sensor values.

Another approach is to use a REST-based standard. We have used the oBIX (Open Building Information Exchange; www.obix.org) [35] standard in some IoT applications, both alone and combined with QLM-MI. oBIX is domain-agnostic enough to be a candidate for general IoT use, however the oBIX Technical Committee states that it does not intend to extend oBIX into other application domains than mechanical and electrical systems in commercial buildings. Technically, oBIX has many similarities with QLM but it lacks several important features. oBIX implements a QLM-like subscription service but it does not support the usage of a callback address and it lacks other features such as TTL and sampling interval specifications. oBIX is only specified for HTTP and WSDL/SOAP protocols, and only oBIX-formatted payloads can be transported. Also, the "piggy backing" functionality is not supported in oBIX (see QLM property 6 in TABLE II), which is a necessary feature for near real-time and two-way communications with mobile nodes and/or nodes located behind a firewall.

VI. CONCLUSION

In the IoT, mobile users and objects will be able to dynamically discover and impromptu interact with heterogeneous physical and virtual resources, as well as virtual data and environments. Nonetheless, based on our experience with the IoT and the different messaging and communication standards related to it, as well as on our experience from creating numerous IoT implementations in different domains, we claim that there remains a true need for sufficiently generic and generally applicable application-level IoT messaging standards. In this paper, we present a minimal set of requirements identified for such IoT messaging standards. The QLM Data Format (QLM-DF) and QLM Messaging Interface (QLM-MI) specifications presented here are candidates for becoming that standard. A major purpose of this paper is to promote QLM standards while giving an opportunity to provide feedback to the people working on the standard or even join the standardization workgroup. The importance of this standardization work for both academic and commercial purposes should not be under-

estimated because the lack of a satisfactory application-level messaging standard is still a great obstacle for realizing the IoT.

Official QLM messaging specifications are expected to be made public by The Open Group in 2014 but several companies and academic organizations are already using the newest specifications for implementing both academic and commercial-level systems. Because QLM-MI allows setting up ad-hoc and flexible communication links that fulfill the requirements of most IoT-like applications, it should be a “safe choice” even for the early adopters. QLM-MI and QLM-DF can be applied to virtually any kind of information, i.e. not only physical products but also to documents, document repositories, and so on. Querying for available design documents, subscribing to the addition/deletion/modification of documents, as well as subscribing to particular change events in design documents is conceptually similar to queries and subscriptions for physical products.

However, creating such standards and getting them into widely use tends to be a long and challenging task, as shown e.g. for the EPC standards in [36]. Indeed, the specification of a “good” standard is not only an engineering task. The current QLM messaging specifications are a result of over ten years of research work jointly with many academic and industrial partners.

REFERENCES

- [1] K. Ashton, “Internet things – MIT, embedded technology and the next internet revolution,” *Baltic Conventions, The Commonwealth Conference and Events Centre*, London, UK, May 2000, pp. 1–5.
- [2] E. A. Lee, “Cyber physical systems: Design challenges,” *11th IEEE International Symposium on Object Oriented Real-Time Distributed Computing*, Orlando, USA, May 2008, pp. 363–369.
- [3] H. Sundmaeker, P. Guillemin, P. Friess, and S. Woelfflé, “Vision and challenges for realising the Internet of Things,” *European Commission*, Brussels, Belgium, 2010.
- [4] L. Baltrunas, B. Ludwig, S. Peer, and F. Ricci, “Context relevance assessment and exploitation in mobile recommender systems,” *Personal and Ubiquitous Computing*, vol. 16, no. 5, pp. 507–526, 2012.
- [5] H. Cao, P. Folan, J. Mascolo, and J. Browne, “RFID in product lifecycle management: a case in the automotive industry,” *International Journal of Computer Integrated Manufacturing*, vol. 22, no. 7, pp. 616–637, 2009.
- [6] D. J. Cook, and J. C. Augusto, and V. R. Jakkula, “Ambient intelligence: Technologies, applications, and opportunities,” *Pervasive and Mobile Computing*, vol. 5, no. 4, pp. 277–298, 2009.
- [7] G. G. Meyer, K. Främling, and J. Holmström, “Intelligent products: A survey,” *Computers in Industry*, vol. 60, no. 3, pp. 137–148, 2009.
- [8] L. Atzori, A. Iera, and G. Morabito, “The internet of things: A survey,” *Computer Networks*, vol. 54, no. 15, pp. 2787–2805, 2010.
- [9] S. Sarma, D. Brock, and D. Engels, “Radio frequency identification and the electronic product code,” *IEEE Micro*, vol. 21, no. 6, pp. 50–54, 2001.
- [10] M. Kärkkäinen, J. Holmström, K. Främling, and K. Artto, “Intelligent products—a step towards a more effective project delivery chain,” *Computers in Industry*, vol. 50, no. 2, pp. 141–151, 2003.
- [11] L. E. Holmquist, H.-W. Gellersen, G. Kortuem, S. Antifakos, F. Michelles, B. Schiele, M. Beigl, and R. Mazé, “Building intelligent environments with smart-Its,” *IEEE Computer Graphics and Applications*, vol. 24, no. 1, pp. 56–64, 2004.
- [12] H. B. Jun, D. Kiritsis, and P. Xirouchakis, “Research issues on closed-loop PLM,” *Computers in Industry*, vol. 58, no. 8, pp. 855–868, 2007.
- [13] D. Kiritsis, “Closed-loop PLM for intelligent products in the era of the Internet of Things,” *Computer-Aided Design*, vol. 43, no. 5, pp. 479–501, 2011.
- [14] D. Kiritsis, A. Bufardi, and P. Xirouchakis, “Research issues on product lifecycle management and information tracking using smart embedded systems,” *Advanced Engineering Informatics*, vol. 17, no. 3, pp. 189–202, 2003.
- [15] S. Haller, S. Karnouskos, and C. Schroth, “The internet of things in an enterprise context,” *Springer*, pp. 14–28, 2008.
- [16] C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos, “Context Aware Computing for The Internet of Things: A Survey,” *IEEE Communications surveys & Tutorials*, no. 99, pp. 1–41, 2013.
- [17] P. Bellavista, A. Corradi, M. Fanelli, and L. Foschini, “A Survey of Context Data Distribution for Mobile Ubiquitous Systems,” *ACM Computing Surveys*, vol. 45, no. 1, pp. 1–49, 2013.
- [18] K. Främling, J. Holmström, J. Loukkola, J. Nyman, and A. Kaustell, “Sustainable PLM through Intelligent Products,” *Engineering Applications of Artificial Intelligence*, vol. 26, no. 2, pp. 789–799, 2013.
- [19] J. Stark, “Product lifecycle management: 21st century paradigm for product realisation,” *Springer*, 2011.
- [20] F. Ameri, and D. Dutta, “Product lifecycle management: closing the knowledge loops,” *Computer-Aided Design and Applications*, vol. 2, no. 5, pp. 577–590, 2005.
- [21] K.-D. Thoben, J. Eschenbächer, and H. Jagdev, “Extended products: evolving traditional product concepts,” *7th international Conference on Concurrent Enterprising Engineering the Knowledge Economy Through Co-operation*, Bremen, Germany, Jun. 2001, pp. 429–439.
- [22] K. Främling, J. Holmström, T. Ala-Risku, and M. Kärkkäinen, “Product agents for handling information about physical objects,” *Report of Laboratory of information processing science series B, TKO-B*, Helsinki University of Technology, Finland, 2003.
- [23] M. Kärkkäinen, T. Ala-Risku, and K. Främling, “The product centric approach: a solution to supply network information management problems?,” *Computers in Industry*, vol. 52, no. 2, pp. 147–159, 2003.
- [24] D. McFarlane, V. Giannikas, A. C. Y. Wong, and M. Harrison, “Product intelligence in industrial control: Theory and practice,” *Annual Reviews in Control*, vol. 37, no. 1, pp. 69–88, 2013.
- [25] K. Främling, and M. Maharjan, “Standardized Communication Between Intelligent Products for the IoT,” *11th IFAC Workshop on Intelligent Manufacturing Systems*, São Paulo, Brazil, May. 2013, pp. 157–162.
- [26] K. Främling, M. Harrison, J. Brusey, and J. Petrow, “Requirements on unique identifiers for managing product lifecycle information: comparison of alternative approaches,” *International Journal of Computer Integrated Manufacturing*, vol. 20, no. 7, pp. 715–726, 2007.
- [27] S. Parrotta, J. Cassina, S. Terzi, M. Taisch, D. Potter, and K. Främling, “Proposal of an Interoperability Standard Supporting PLM and Knowledge Sharing,” *Journal*, Springer, 2013.
- [28] E. Huvio, J. Grönvall, and K. Främling, “Tracking and tracing parcels using a distributed computing approach,” *Proceedings of the 14th Annual conference for Nordic researchers in logistics*, Trondheim, Norway, Jun. 2002, pp. 29–43.
- [29] K. Främling, T. Ala-Risku, M. Kärkkäinen, and Holmström, J., “Design patterns for managing product life cycle information,” *Communications of the ACM*, vol. 50, no. 6, pp. 75–79, 2007.
- [30] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, “Design patterns: elements of reusable object-oriented software,” *Reading: Addison Wesley Publishing Company*, 1995.
- [31] P. T. Eugster, P. A. Felber, R. Guerraoui, and A.-M. Kermarrec, “The many faces of publish/subscribe,” *ACM Computing Surveys*, vol. 35, no. 2, pp. 114–131, 2003.
- [32] R. T. Fielding, “Architectural styles and the design of network-based software architectures,” *PhD thesis: University of California*, Information and Computer Science, 2000.
- [33] S. Kubler, M. Madhikermi, and K. Främling, “QLM messaging standards: introduction and comparison with existing messaging protocols,” *Service Orientation in Holonic and Multi-Agent Manufacturing and Robotics*, Springer International, vol. 544, pp. 237–256.
- [34] M. Hapner, R. Burrige, R. Sharma, J. Fialli, and K. Stout, “Java message service,” *Sun Microsystems Inc., Santa Clara, CA*, 2002.
- [35] M. Neugschwandtner, G. Neugschwandtner, and W. Kastner, “Web services in building automation: Mapping KNX to oBIX,” *5th IEEE International Conference on Industrial Informatics*, Vienna, Austria, Jul. 2007, pp. 87–92.
- [36] K. Främling, V. Hinkka, S. Parmar, and J. Taitilä, “Assessment of Standards for Inter-Organizational Tracking Information Exchange in the Supply Chain Information Control Problems in Manufacturing,” *Information Control Problems in Manufacturing*, Bucharest, Romania, May 2012, vol. 14, pp. 661–666.