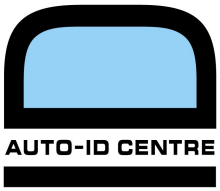
Published June 1, 2003. Distribution restricted to Sponsors until September 1, 2003.



WHITE PAPER

PML Server Developments

Mark Harrison, Humberto Moran, James Brusey, Duncan McFarlane

AUTO-ID CENTRE INSTITUTE FOR MANUFACTURING, UNIVERSITY OF CAMBRIDGE, MILL LANE, CAMBRIDGE, CB2 1RX, UNITED KINGDOM

ABSTRACT

This paper extends our previous white paper on our PML Server prototype work. We begin with a brief review of the Auto-ID infrastructure, then consider the different types of essential data which could be stored about a tagged physical object or which relate to it. In our data model we distinguish between data properties at product-class level and at instance-level. Product-class properties such as mass, dimensions, handling instructions apply to all instances of the product class and therefore need only be stored once per product class, using a product-level EPC™ class as the lookup key. Instance-level properties such as expiry date and tracking history are potentially unique for each instance or item and are logically accessed using the full serialised EPC™ as the lookup key. We then discuss how a PML Service may use data binding tools to interface with existing business information systems to access other properties about an object besides the history of RFID read events which were generated by the Auto-ID infrastructure. The penultimate section analyses complex queries such as product recalls and how these should be handled by the client as a sequence of simpler sub-queries directed at various PML services across the supply chain. Finally, we introduce the idea of a registry to co-ordinate the fragmented PML Services on a supply chain in order to perform tracking and tracing more efficiently and facilitate a complex query, which requires iterative access to multiple PML Services in order to complete it.

WHITE PAPER

PML Server Developments

Biographies

Mark Harrison
Senior Research Associate

Mark Harrison is a Senior Research Associate at the Auto-ID Centre lab in Cambridge working on the development of a PML server, web-based graphical control interfaces and manufacturing recipe transformation ideas. In 1995, after completing his PhD research at the Cavendish Laboratory, University of Cambridge on the spectroscopy of semiconducting polymers, Mark continued to study these materials further while a Research Fellow at St. John's College, Cambridge and during 18 months at the Philipps University, Marburg, Germany. In April 1999, he returned to Cambridge, where he has worked for three years as a software engineer for Cambridge Advanced Electronics/ Internet-Extra, developing internet applications for collaborative working, infrastructure for a data synchronisation service and various automated web navigation/capture tools. He has also developed intranet applications for his former research group in the Physics department and for an EU R&D network on flat panel displays.

Humberto J Moran
Senior Research Associate

With more than ten years of experience in information system implementation, Humberto Moran has occupied relevant positions in leading corporations such as Unisys, Lafarge, Oracle, and his own entrepreneurial venture. He has studied Computer Engineering – "Universidad Simon Bolivar"/Caracas; a Ph.D. in International Economics – "Universidad Complutense de Madrid"/Madrid; and an MBA in the Judge Institute of Management, University of Cambridge.

WHITE PAPER

PML Server Developments

Biographies

James Brusey
Senior Research Associate

James Brusey previously worked (for about 13 years) in computer system administration, specialising in IBM mainframe assembler. He received a B.Ap.Sci in Computer Science from RMIT University (Melbourne, Australia) in 1996. He began studying autonomous robot control in 1998 and was a team member and the main software developer for RMIT University's Formula 2000 RoboCup team, which made the finals in the 2000 games.

James' Ph.D. is entitled "Reinforcement Learning for Robot Soccer". It developed a novel approach to bootstrapping reinforcement learning and also examined simulation-based reinforcement learning for a real robot.

Duncan McFarlane
Research Director Europe

Duncan McFarlane is a Senior Lecturer in Manufacturing Engineering in the Cambridge University Engineering Department. He has been involved in the design and operation of manufacturing and control systems for over fifteen years. He completed a Bachelor of Engineering degree at Melbourne University in 1984, a PhD in the control system design at Cambridge in 1988, and worked industrially with BHP Australia in engineering and research positions between 1980 and 1994. Dr McFarlane joined the Department of Engineering at Cambridge in 1995 where his work is focused in the areas of response and agility strategies for manufacturing businesses, distributed (holonic) factory automation and control, and integration of manufacturing information systems. He is particularly interested in the interface between production automation systems and manufacturing business processes.

WHITE PAPER

PML Server Developments

Contents

1. BACKGROUND

The key to the Auto-ID architecture [1] is the Electronic Product Code (EPC™) [2], which extends the granularity of identity data far beyond that which is currently achieved by most bar code systems in use today. The EPC™ contains not only the numeric IDs of the manufacturer and product type (also known as stock-keeping unit or SKU) but also a serial number for each item or **instance** of a particular product type. Whereas two apparently identical instances or items of the same product type may today have the same bar code, they will in future have subtly different EPCs™, which allows each one to have a unique identity and to be tracked independently.

In order to minimise the costs of Radio Frequency Identification (RFID) tags [3], the Auto-ID Centre advocates that only a minimal amount of data (the EPC™) should be stored on the tag itself, while the remaining data about a tagged object should be held on a networked database, with the Electronic Product Code™ (EPC™) being used as a database key to look up the data about a particular tagged object. Within the Auto-ID infrastructure, the Savant™ [4], Object Name Service (ONS) [5] and PML Service [6] are all networked databases of some form.

Edge Savants™ interface directly with RFID readers and other sensors and generate Auto-ID event data, typically consisting of triples of three values (Reader EPC™, Tag EPC™, Timestamp) and an indication of whether the tag has been 'added' or 'removed' from the field of the tag readers.

The Object Name Service (ONS) is an extension of the internet Domain Name Service (DNS) [7] and provides a lookup service to translate an EPC™ number into an internet address where the data can be accessed.

Data about the tagged object is communicated using the Physical Markup Language (PML) [8,9] and the PML Service provides additional information about the tagged object from network databases. The Physical Markup Language (PML) does not specify how the data should be stored, only how it should be communicated. It should be possible for many different types of existing information systems to act as data sources to the PML Service, and for the data to be queried and communicated using the PML language and by reference to the PML schema [10] rather than by reference to the particular structure/schema of the various underlying databases in which the values are actually stored. We described these ideas in a previous white paper [6].

In Section 3 we discuss how data binding tools may allow the PML Service to interface with existing information systems.

Because of security and privacy concerns, the PML Service will be fragmented across the supply chain, with each party handling the object being responsible for updating their own internal databases with information about the tagged object while it is in their custody. Users of the PML Service (clients) outside the company will only have read-only access to a controlled subset of the data, by querying the company's PML service. The company providing the PML service will be able to define precisely which subset of data each authenticated user or group of users is entitled to access, depending on the business relationship with the company (e.g. retail partner, logistics partner, etc.).

We envisage that the data content will be derived from both existing business information systems as well as new Auto-ID tracking event data. The security layer of the PML Service allows each company to control exactly which subsets of their internal data are exposed through the PML Service, while the authentication layer of the PML Service allows them to tie different subsets of their data to different users of the PML Service.

Certain complex queries, such as a product recall may need to access fragments of PML data from multiple servers across the supply chain. In Section 5, we discuss how we might use a registry to co-ordinate the navigation between PML servers to facilitate such complex queries.

We now discuss types of data related to tagged objects and how this classification helps when analysing complex queries which operate on multiple PML Services.

2. TYPES OF DATA

Besides the different database structures (e.g. XML [11] product catalogs, relational databases (SQL) [12] and directory services), in which the underlying data may be physically stored, there are several fundamentally different types of data associated with an object and also several types of queries we might want to make. Table 1 provides examples of a number of types of data related to an object.

Table 1: Types of data related to tagged objects

DATA TYPE	EXPLANATION	UPDATE FREQUENCY	HOW DISTRIBUTED	# OF KEYS	DATA SOURCE
Class-level Manufacturer's data	a.k.a. 'Product-level data' standard properties and procedures applying to all instances of the same product type e.g. mass, dimensions, instructions on safety, handling, care, usage etc.	fairly static	single server	small	product catalog data
Serial-level Manufacturer's data	Instance-specific data which either – is not defined at product-level – overrides values defined at product-level e.g. lot control, expiry date, customisation parameters	fairly static but updated for different batches	single server	large (up to one key per instance per product)	existing business information systems – instance-level access may be lacking
Serial-level tracking/ sensor data	object-centric track and trace data 'Where was the object O at time T?' e.g. (location, time, identity) – Auto-ID and sensory event data + timestamped sensor data (e.g. temperature, humidity...) – changes of custody – change of aggregation state (packing, palletisation, embedding within a compound/composite product) – change of EPC™ to track (e.g. track pallet super-tag	very dynamic	multiple servers + registries?	very large (up to one per key per instance per product type per custodian)	Savant™
Transactions	business documents which refer to EPCs™ e.g. advance shipping notices, purchase orders bills of lading etc.	fairly static	multiple existing business information systems	moderate key is the document ID – but the document itself may refer to multiple EPC™s	existing business information systems, as data source and persistent repository
Location-centric data	which tag EPCs™ were seen by reader R at location X in time range T1–T2?	very dynamic	multiple reader locations – one server per cluster/building?	one key per reader EPC™	Savant™

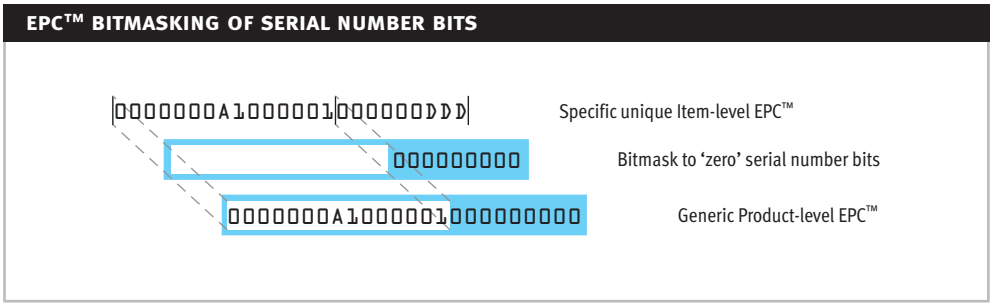
In some cases, we want to obtain properties that are common to all unique instances of the product class. An example may be the mass or dimensions. In other cases, we may want to access properties whose values differ for different instances of the product – e.g. the expiry date. For improved visibility of the supply chain, we certainly need to be able to access instance-level tracking data – i.e. the history of the tagged object – which readers it passed and when.

As we discussed in our previous paper, we envisage that simple PML queries will use two parameters:

1. the EPC™ – identifies the tagged object of interest
2. a pathname which identifies the property of interest within the PML Schema. The pathname may be an XQL [13] or XPath [14] expression or a Java bean accessor method [e.g. getMass()] [15]

If we specify the full EPC™, including the serial number bits, a simple query should return the property if it is defined in the instance-level data. If the property is not defined in the instance-level data, the PML Service should automatically perform a bitmask operation on the EPC™ to set the serial number bits to zero, as illustrated in Figure 1, then retry the query. In this case, the property defined at the product-class level would be returned instead.

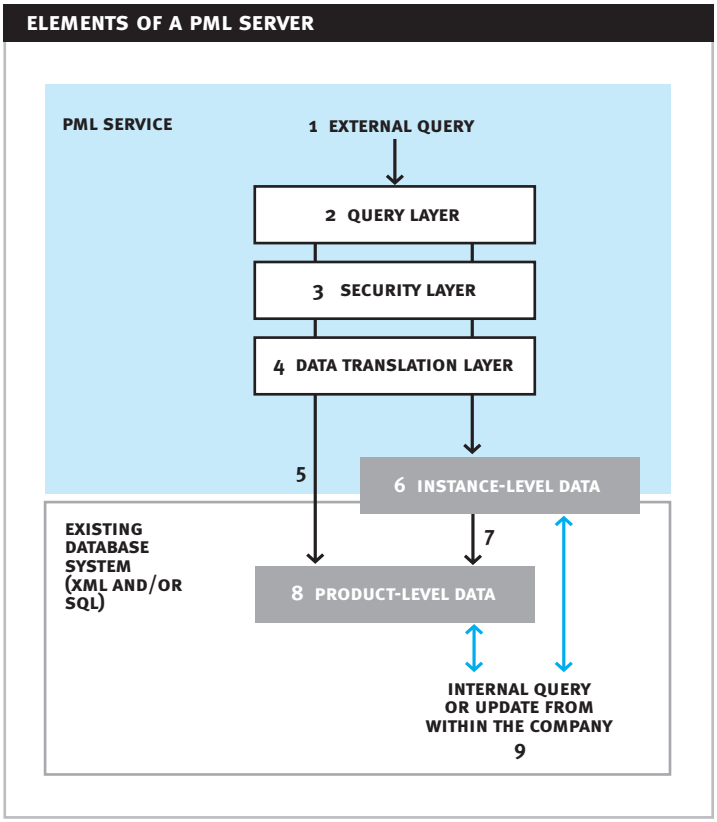
Figure 1: Bitmasking of an instance-level Type I EPC™ to obtain an EPC™ class for all items of that product class.



Of course, the client could have supplied a bitmasked class-level EPC™ in the first place if the property is known or considered to be the same for all instances of the product. This 'fall-through' from instance-level data to the corresponding product-class data is illustrated in Figure 2.

Figure 2: Schematic structure of a PML server, indicating 'fall-through' to return product-level data where a particular property is not defined at instance-level.

1 From outside the company
2 Returns PML subset or fragment given an EPC™ and XML Path or Java bean accessor method
3 (checks user's access rights to a particular subset of the PML data tree
4 (maps XML & RDBMS to PML Schema) (– both the query and the returned data)
5 Product-level data can be accessed directly if a product-level EPC™ is specified
6 XML or Relational Database Tables
7 'Fall-through' to product-level data where instance-level data is not defined
8 XML or Relational Database Tables
9 (including data from Savant™) (using existing SQL methods)



We suggest that the class-level EPC™ (see Figure 1) referring to all instances/items of a particular product class should have the same number of digits as the corresponding instance-level serialised EPC™, rather than being represented as a truncated EPC™ prefix. This avoids any ambiguity about whether a truncated EPC™ prefix should be left-padded or right-padded with zeros or which ranges of bits correspond to which data partition and has the advantage that the class-level EPC™ could be stored in the same database column or field as a regular serialised instance-level EPC™. Furthermore, the use of all zeroes in the serial number partition clearly marks the class-level EPC™ as being special and distinct from instance- level EPCs™; conversely, the serial number consisting of all bits set to zero should be reserved for this use – to indicate or access product-level data; serial numbers should therefore begin incrementing from 1 upwards – **not** from zero upwards.

Not all of our queries will use the object's EPC™ as the database key. Sometimes our query uses another lookup parameter, such as a particular transaction number or a particular location or reader EPC™ and obtains a list of EPCs™ as the response.

We may have the transaction ID of a particular business document, such as a purchase order or advance shipping notice, which itself refers to or contains several EPCs™ within the document.

We may also have to deal with location-centric queries, e.g. in the case of a product recall, once we have established the location where the defects or contamination occurred, we then want to be able to query the reader at that location for all EPCs™ which it observed until the problem was rectified, so that we can then trace their current location and alert the retail outlets and logistics companies.

Table 1 attempts to classify some of these fundamental data types and also provides some indications about how static or dynamic each type of data may be, as well as how many separate servers may hold such data and how many keys are needed to access the data. Such considerations are a factor in the determination of the appropriate type of underlying database for persistent storage of the data.

3. DATA BINDING TOOLS FOR A PML SERVICE

The PML Service is intended to be a common global interface for queries about tagged physical objects. Auto-ID produces a large volume of new Auto-ID event data with instance-level tracking granularity. In many cases, this may require companies to re-evaluate their current database architecture, especially if they intend to archive such Auto-ID event data for auditing and tracing purposes into the future.

EPC™ data from readers is filtered by low-level Savants™ and is then pushed out to either higher-level internal Savants™ or other databases for longer-term storage.

Object-centric PML Services should be able to provide the tracking history for a particular object, in terms of the reader EPCs™ and timestamps as the object passed through a particular custodian's part of the supply chain. Object-centric PML Services answer the question 'where was object O in the time range T1 – T2?'

Location-centric PML Services group the Auto-ID event data for particular readers or locations and instead provide an observation history of the tag EPCs™ observed and the timestamps at which they were added or removed from the reader's field. Location-centric PML Services answer the question 'which objects were seen at location L/by reader R in the time range T1 – T2?'

Furthermore, the object-centric PML Service may also provide other types of common data about an object, drawn from other existing information systems as their data sources. It is not intended that the

Physical Markup Language should necessarily subsume all other markup languages developed for specific industry sectors [16] – but rather that it initially focuses on Auto-ID event data and a limited number of fundamental properties which are essential for supply chain logistics, shipping and storage management. Such properties may include the mass, dimensions, description, safety/handling instructions, date of manufacture, expiry date, etc.

While the Auto-ID event data is well defined in terms of the PML Core schema and originates primarily from systems implementing the Savant™ interface, the other types of object-centric data come from a variety of database types and vendors.

If the PML Service is to provide accurate, up-to-date information, while still presenting a global common interface for queries, the PML Service implementation needs to use data binding tools [17] to connect the other data sources to the PML interface in real time. This allows the companies to continue to use their existing information systems internally, just as before, while the PML Service uses data binding tools to extract particular types of data from these existing systems, without revealing the addresses, connections or internal structure of those sources to the authenticated end-user client of the PML Service. For security or efficiency reasons, the PML Service may actually interface to internal replicas of the operational databases, rather than directly to the main operational databases of a company, although it should be remembered that replica databases may not be as up-to-date as the original operational databases.

Fortunately, sophisticated data binding tools already exist. Castor (an open source data binding project from Exolab.org) [18] provides data binding between Java bean objects and either XML representation, or persistence in relational databases (RDBMS/SQL) or LDAP [19] directory services. JAXB [20] and JDO [21], free data binding tools from Sun Microsystems, also provide similar capabilities. Data binding tools make use of XML Schema [10] to generate Java bean objects [15], which can have a hierarchical set of properties which can be queried or updated using get( ) and set( ) methods.

Database connector files can be used to specify which relational databases to connect to, and which database user-ID to use for login. Mapping files describe the mapping between particular Java bean classes and their properties and the corresponding tables and fields (column names) in a relational database structure. Mapping files can also be used to override the default XML <-> Java bean mapping, for example where inline attributes of elements are used in preference to child elements in the XML (or PML) markup.

Therefore, data binding tools allow each company to control precisely which database tables and fields are exposed for the PML Service, to set up particular database users with read-only access to specific tables and fields, so that different access rights or views of the data can be given to different authenticated users, depending on the business relationship they have with that company.

In addition to the security restrictions which the data binding tools can impose, further access control tools such as XACL [22] and XACML [23] allow companies to precisely define access control rules and policies, and associate these policies with particular users, depending on their role or relationship.

4. PML SERVICE APPLICATIONS AND COMPLEX QUERIES

A company may provide its own internal staff and systems with a local internal PML Service which stores relevant cached data obtained from remote PML Services outside the company, much as a web proxy server or cache stores frequently requested files to reduce the amount of external traffic.

However, the external PML Service provided by a company will usually only provide data which is held by that company. The service will not normally be expected to masquerade as the client and obtain data from other remote PML Services in order to more fully answer the query from a client. It will usually be the responsibility of the client or client application to iteratively contact other servers for further data. The client application is to a PML Service the equivalent of a web browser interacting with a webserver. A web browser displays a composite image of a web page, although some of the data may have been obtained from web servers other than the one specified in the browser's address bar. For example, the HTML markup [24] of a news page on a website may use references to include image files from another source website. When the client application notices that the server it initially contacts is not able to provide all of the information required for the composite view, it iteratively contacts the other servers holding the additional data, in order to build the composite view for the user.

Client applications of PML Services are likely to need to behave similarly and may also need to iteratively issue sub-queries to multiple PML servers across the supply chain in order to obtain all the data required to answer complex queries. However, whereas HTML is a display markup language and the person who writes the web page has a particular composite display in mind, PML is a data exchange markup language and no server has any notion of displaying a comprehensive 'composite view' of all the PML data, nor any motivation to do so. In fact, it is preferable that they supply as little data as possible, returning only the PML fragment required.

Furthermore, the majority of PML data is likely to be available only to particular groups of trusted authenticated users. Each client may use different authentication tokens (user-IDs, digital certificates etc.) to log into different PML Services, showing each PML Service only the appropriate authentication token and hiding the others. Therefore, each service would be unable to masquerade as the client to obtain additional data from other services outside of that company, since only the client retains the authentication tokens for accessing that data. We see that also from this authentication perspective, only the client can build the composite view or perform the complex query using all the data that the client is allowed to see.

Table 2 lists a number of queries which might be asked, outlining the query parameters and expected results, as well as the appropriate service to query at each stage. Complex queries are considered as an iterative sequence of sub-queries directed to the appropriate services in turn.

Table 2: Types of queries for PML Services

QUERY	PARAMETER SUPPLIED TO QUERY	DATA REQUIRED FROM QUERY	PML SERVICE TO QUERY
What sort of object is this EPC™?	Product-class EPC™ **or** Instance-level EPC™	product-class description **unless** superseded by instance-level description	manufacturer's public-domain PML descriptions
Where is my object now?	Instance-level EPC™	ID of latest custodian	navigation to custodians **then** PML service of last custodian
Which path did it take?	Instance-level EPC™	sequential list of IDs of all custodians	navigation to custodians **and optionally** PML services of custodians
Where was it delayed?	Instance-level EPC™	sequential list of IDs of all custodians with arrival/ departure timestamps	navigation to custodians **then** internal trace history within a particular custodian's data
I need to calculate shipping costs. How much does it weigh?	Product-class EPC™ **or** Instance-level EPC™	product-class mass **unless superseded** by instance-level mass	manufacturer's public-domain PML descriptions
I need to allocate storage space. What are its dimensions?	Product-class EPC™ **or** Instance-level EPC™	product-class dimensions **unless superseded** by instance-level dimensions	manufacturer's public-domain PML descriptions
My object was contaminated. Where did this happen and which other types of product crossed paths or were co-located with my object?	Instance-level EPC™ **then** Reader EPCs™, timestamp ranges **then** 'foreign' EPCs™ at Instance-level or Product-class level	sequential timestamped list of all reader EPCs™ which saw my object **then for each reader** list of other EPCs™ seen within the same time range **then for each 'foreign' EPC™** description of product type of 'foreign' EPC™	navigation to custodians **then** location-centric PML services **then** manufacturers' PML descriptions of products which crossed paths
We know that all objects which passed through location L in time range T1–T2 may be defective. Where are they now?	Reader EPCs™, timestamp ranges **then** Instance-level EPC™	list of all tag EPCs™ which were seen by those readers **then for each tag EPC™** sequential list of subsequent custodians	location-centric PML services **then** navigation to subsequent custodians

5. CO-ORDINATING MULTIPLE SERVERS ACROSS THE SUPPLY CHAIN

The analogy with the web browser and webserver is useful, although it must be remembered that there is a clear difference for the PML Service. Someone designs a web page with a composite view in mind for a human observer to read, even though the elements of the page may be drawn from multiple servers. The PML Service is not designed to produce such a composite view. Unlike a static web page, the response from a PML Service depends on the query which was asked and typically consists of just a particular relevant fragment of the whole PML data which exists about a given object, in order to minimise bandwidth usage and reduce the need for the client to parse vast amounts of markup data when only a small fragment is required as the response.

Since the PML Service has neither the capability nor any self-interest to provide a composite view of all the data for a human observer, the client application needs to make use of the object name service (ONS) to locate at least one network address, then use some additional means for navigating across the supply chain to find all the PML Services which may hold the required data about a given EPC™.

The ONS is built on the Domain Name Service and converts an EPC™ number into an IP address. However, efficiency and scalability require that it does so algorithmically, keeping the number of unique database keys to a minimum, rather than keeping a separate record for each of the several trillion items produced each year, which would rapidly slow down the database lookup. The Domain Name Service (DNS) upon which ONS is built, is robust but not really intended for dynamic updating, since updates to DNS itself typically take a number of days to propagate across the internet until all other DNS servers worldwide reflect the update. Furthermore, a typical DNS entry points to a particular primary IP address or cluster of addresses, sometimes with secondary addresses or tertiary addresses for backup or when the primary address is unreachable, with the secondary or tertiary addresses usually providing replicas or mirrors of the same information held at the primary address.

The situation of PML data fragmented across the supply chain is somewhat different, since although each company may hold some data relevant to a particular EPC™, each holds a **different set** of data, collected while the object was in **their** custody, rather than each holding a replica of each other's data. Furthermore, supply chain logistics are likely to become increasingly dynamic and flexible, so that the logistics paths along the supply chain are neither fixed nor pre-determined but use the benefit of real-time high-granularity Auto-ID data correlated with other sensory data to reschedule and re-route in real time, either to meet the demands of the customer and deal with rush orders, or to ensure that perishable goods reach customers in optimal condition.

The ONS provides a scalable lookup service but it is not intended that its records will by dynamically modified as objects pass from one company to another along the supply chain, although it is clearly necessary that something else provides real-time dynamically updateable directions to all the various PML Services which may hold data about the object.

So long as the tag remains attached to the object and the EPC™ transmitted by the tag is immutable, then the ONS will almost always point to one particular address throughout the life of the object, or at least up to point-of-sale, when the tag may be deactivated. We call the address pointed to by the ONS the primary address. This means that we must have another mechanism for recording the times when the custody of the object changed or when an object undergoes aggregation or disaggregation.

The essential data which is needed to record change of custody is:

- Object ID (EPC™)
- Arrival or release time
- Identity of the next custodian of the object (either as a virtual EPC™ [25] or network address)

When one object is combined with another object to assemble a higher-level composite product with a different EPC™ tag, we can consider this as a form of aggregation. The reverse process, breaking down a composite object into smaller components is a form of disaggregation. Just as it is important to record change of custody, it is also important to record the next EPC™(s) to track after an aggregation/disaggregation procedure. The data which is needed to continue tracking an item after aggregation or disaggregation is:

- Object ID (EPC™)
- Time of assembly or disassembly
- A list of the next EPCs™ to track – either the composite product or the broken-down components

When objects are deliberately aggregated into a pallet or container or broken down into smaller units, verification and subsequent recording of this change of containment in the data should reduce the potential for errors when inferring from readings of only some of the tags present in the aggregate or container.

So far, we can conceive of two alternative methods by which such changes of custody or containment can be recorded. Both have potential advantages and disadvantages:

The first method is for each server to simply provide forward and reverse links embedded within their own PML data to point to the previous and next custodian, who may also hold relevant data on that EPC™. It would then be possible to trace the movement of the object along the supply chain by daisy-chaining these together, contacting each PML Service in sequence until the end of the chain is reached, indicating the current custodian. This approach may appear to be more secure, since there is no central registry holding data about flows of goods, which is commercially sensitive information. However, this method is less efficient for the client, since the client needs to interrogate each server in turn just to navigate along the supply chain for basic tracking and trace applications. Furthermore, the method is not robust when one of the servers on the chain fails to provide onward directions, either due to temporary network connectivity problems or as a result of a political disagreement between parties on the chain.

Our preferred approach is to use a secure registry to hold a real-time updateable sequential record of all of the custodians which have handled a particular EPC™, together with their identity or internet address and perhaps also the timestamps of when the object arrived with them and when it subsequently left their custody. This approach overcomes the problem of a weak link in the former approach, although this means that the registry clearly needs to be resilient (i.e. replicated and probably also distributed) and secure, providing the navigational links to valid authenticated users only on a 'need to know' basis. Each custodian handling the object would have authorisation to send an update record to the registry to notify the arrival or release of the tagged object. To reduce bandwidth usage, it should be possible for the update message to contain several EPCs™ of the same product type, where they were all received or despatched at the same time.

This method should also provide much faster tracking and tracing capabilities, since it reduces the need to contact several PML servers individually to determine the current custodian of an object or the path taken. The registry can supply this information directly. Indeed, the registry could even provide tracking or trace functions as part of its core functionality. Both functions would take an EPC™ as the input parameter. The trace function could return an audit trail consisting of a sequential list of network addresses for the PML services of the custodians and the corresponding EPCs™ to query at each custodian. The tracking function would merely provide the last entry of this list, to indicate the current or last custodian of the tagged physical object.

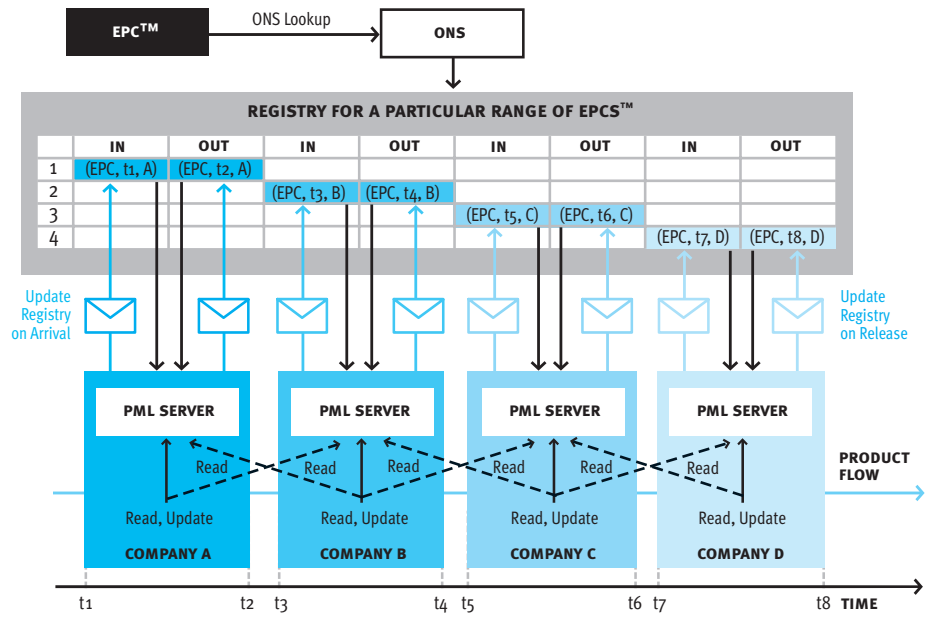
Finally, the client can choose to contact the last custodian in the sequence to obtain more detailed spatial information, such as which shelf of a retail store holds the object. It seems logical to consider the registry as a separate entity from that of the PML Service, since it holds very specific information about arrivals, departures, custodian addresses and could be implemented with a single relational database as its persistent storage and a small number of well-defined methods for efficiently performing tracking and tracing operations on an EPC™ specified by the client. Where a registry approach is deployed to co-ordinate PML services in a particular supply chain, it may be more appropriate for the network address returned by an ONS lookup of an EPC™ to point directly to the address of the registry, rather than the first PML service on that chain. We illustrate this approach in Figure 3. The issue of who owns or should pay for the secure registry lies outside the scope of this paper; this may vary for different industry sectors, especially for those industries which are highly regulated by government or require very high levels of traceability.

Figure 3: Simplified network diagram showing how a registry could be used to co-ordinate navigation between PML services on a supply chain. Each registry would handle a particular range of EPCs and would be pointed to by the ONS lookup. Each party on the chain would send a short secure message to the registry to notify receipt or release of the tagged object(s). The registry could then be queried on a 'need-to-know' basis to determine who is the current custodian, which path was taken by the object, as well as facilitating product recalls.

✉ = Data packet:

- EPC™
- Arrival/release time
- Custodian ID
- EPC™ to track in future (Empty Schema of type of data stored)

CO-ORDINATING PML SERVICES ACROSS A SUPPLY CHAIN USING A REGISTRY



Apart from tracking and tracing applications, product recalls and return logistics are probably the most complex, demanding types of query which may need access across the whole of the supply chain. Even within product recalls, there will be differences between industry sectors about the urgency of a product recall – whether the defect or contamination is potentially hazardous or life-threatening, in the case of the food and pharmaceutical industries – or has merely resulted in a functionally defective product where the defect itself poses no immediate danger, but the product fails to meet its intended purpose, specifications or expected durability. Such considerations, together with regulatory legislation will determine which method of navigating along the supply chain is acceptable, balancing the urgency of the recall against commercial sensitivity about a third-party registry holding sensitive data about flows of goods.

We do not preclude that the details listed above could also be stored locally by each company, or that they could appear within the PML markup for the data. However, we advocate that the registry should be separate from the PML services, since the registry is updated by several approved parties, whereas the PML service usually provides only a read-only view to clients outside the company. We also recommend validation of the incoming registry updates from third parties, to check that the authenticated sender has authority to access the registry and also to verify that the data is as expected – i.e. that its parameters contain an EPC™, an arrival/release time and the indentifier of the next custodian, all in the correct data type and format.

The registry allows us to know which party to contact to obtain the PML data stored about an object within a given timeframe. A further refinement would be to know which party to contact about a particular type of property or measurement of the object, such as its temperature. Technically, this could be achieved if upon release of the object, each party sends an empty PML Schema along with the release time etc. back to the registry. The empty PML Schema they send would not contain any data values but could indicate which fields of the data they had updated and would therefore give authorised users of the registry an immediate indication of which servers to contact directly in order to obtain that data, rather than having to query each one in turn. There may be commercial reasons why companies may prefer not to publish (some parts of) the empty PML schema representing the nature of the data they hold about the object – but that discussion lies outside the scope of this paper.

Our preferred model for coordinating a supply chain of PML servers and joining together the data fragments when the need arises is to maintain a separate registry whose network address is indicated by the ONS lookup of the object's EPC™. The registry should always be reachable and for each EPC™ it handles, it should hold sequential records of arrival time, release time and next custodian, to facilitate complex, iterative queries where necessary. From a technical perspective, storage of empty PML schemas provided by each custodian upon release of the object would facilitate identification of which party to contact for data of a particular type, as well as within a particular timeframe.

We consider that the registry offers several benefits when co-ordinating a number of related PML services in the supply chain, as listed in Table 3.

Table 3: Benefits of the proposed registry model for co-ordinating multiple PML services in a supply chain, where each PML service only holds a fragment of the object's PML data, collected while the object was in their custody.

ADVANTAGES OF A CO-ORDINATION REGISTRY	
EFFICIENCY	By maintaining a registry to maintain knowledge of the time interval when each company had custody of the object and potentially also store the types of properties held at each member of the chain, we avoid the need to query each server in turn just to check if they hold the data required by our query. Track and trace could be core functionality provided by the registry.
ROBUSTNESS	The registry points to each member of the chain. This is more robust than if each link on the chain only maintained a forward and reverse link to its immediate neighbours, since navigation along the chain would be interrupted if any of the intermediate links of the chain were unreachable.
FLEXIBILITY	The registry does not prescribe who (e.g. manufacturer, retailer, third party) must host it. The choice of the EPC number determines this, since the registry is whichever network address is pointed to by the ONS lookup of the EPC. The protocol we propose is identical for all members of the chain – they notify the registry on arrival/release of an object and otherwise perform their actions on the object, updating their own records about the object while it is in their custody. Another aspect of the flexibility is that companies can continue to use their existing relational databases, with the addition of extra tables and fields for recording instance-level data. They can continue using the well established SQL protocol for querying and updating their own data. The additional infrastructure required for the PML service is predominantly an XML-based interface for dealing with external queries and mapping these to controlled portions of the database.
SECURITY	The registry avoids any need for companies to update each other's PML data – they continue as at present, updating their own records about the product data and merely send a data packet with arrival/release timestamps to the registry to facilitate traceability along the chain. All external requests to PML servers are by definition queries rather than updates.
RELIABILITY	So long as the registry for each chain is guaranteed to be always online, having redundant servers, alternative network connectivity and backup power supplies, then queries across the supply chain can continue. If one of the other PML servers is temporarily unavailable, then either previously cached data can be used and/or the query can be queued for retry later.

6. USE OF A PML SERVICE REGISTRY FOR PRO-ACTIVE TRIGGERING OF PRODUCT RECALLS

The much finer granularity of instance-level product data together with real-time object tracking enabled by Auto-ID technology should facilitate more rapid product recall where the need arises, together with marked reductions in wastage since the unsafe goods can be identified individually by their EPCs™ rather than by the batch ID of a particular pallet or case. Meanwhile, the instance-level tracking history data should assist with diagnosis of why the product recall was necessary and when and where the problem (e.g. contamination, defective processing, unsafe storage temperature) occurred.

In the previous section we proposed an approach in which a registry (as identified by the ONS lookup of the EPC™) is updated upon arrival and release of the object. Arrival and release times are very important opportunities for checking whether the products being handed over are already marked for product recall and if so, triggering an alert and invoking an alternative course of action for the problem objects, once they have been identified and removed.

We propose that a logical place to record that a particular product is marked for recall is in a dedicated database table within the registry which handles that EPC™ range. With the EPC™ or an EPC™ pattern acting as the database key, a simple one-bit field could be used to flag an alert, while a further field could provide additional information, such as the address of a web page for further information. If this product recall database table were automatically queried at the same time as the registry is being updated with the arrival/release times, then we have the opportunity to use the response of the incoming registry update to send a return value listing any EPCs™ which are marked for product recall and providing the web addresses for additional information. Further enhancements are possible, so that the additional information is customised to provide relevant (context-sensitive) information on how to deal with the object's product recall, depending on the party's role in the supply chain, whether a supplier, manufacturer, distributor or retailer.

7. SUMMARY

- The PML Service provides clients with access to data about tagged objects by using an open, global markup language and a hierarchical PML schema.

- Clients can query the PML Service using an EPC™ to identify a tagged object and a pathname (or equivalent Java bean accessor method) to specify the property of interest within the PML schema.

- The PML Service should be able to supply data from various sources, including Auto-ID event data from Savant™ and product data from existing information systems, all using the PML schema interface, thus hiding the internal connections and structure of the underlying database sources from the client using the PML Service.

- PML Service implementations may use data binding technologies such as Castor, JAXB, JDO to connect the PML Service interface to existing corporate data sources and to restrict the data exposed via the PML Service to a subset of their internal data, which they are prepared to release to trusted partners.

- The PML Service will generally support authenticated access but may also support public or anonymous access. The PML Service should be able to differentiate between authenticated users on the basis of groups or roles reflecting their relationship with the company and should be able to expose a different subset of data to them, accordingly.

– Each company which handles a tagged object is expected to update records in their own data systems about location tracking, measurements and actions performed on the object while it is in their custody. The PML Service is used to selectively expose a company-defined subset of these records to a particular trusted partner.

– The external PML Service exposed by a company will normally be read-only to clients outside the company, so external updates will not normally be allowed through the PML Service interface.

– The comprehensive PML data about a tagged object will not normally exist on a single server, but will instead be fragmented across several parties on the supply chain who have handled the tagged object – we call these custodians.

– The number and sequence of custodians for a given EPC™ will change over time more rapidly than typical changes to DNS entries take to propagate. Furthermore, ONS is not intended to be updated frequently.

– For robustness, we advocate the use of a secure registry to hold a sequential list of all custodians for a given EPC™. This will greatly facilitate track and trace applications, as well as more complex queries needed for product recalls etc.

– Complex queries may require the client to iteratively send sub-queries to multiple PML Services and the registry, as necessary, each time authenticating with the appropriate authentication token (e.g. user-ID, digital certificate).

– The PML Service is not normally expected to perform complex iterative queries on behalf of a client, nor to masquerade as that client in order to obtain additional data from other services on behalf of the client making the initial query.

– Performing complex queries remains the role of the client – not the PML Service. It suffices that a company's PML Service merely provides access to the subset of the company's data about the tagged object, which the company is prepared to expose to its authenticated trusted partners.

– The registry may also be an appropriate place to store flags to indicate that a product should be recalled or to store metadata or an empty PML schema to indicate the type of data provided by each PML Service.

8. ACKNOWLEDGMENTS

We are grateful for fruitful discussions with our colleagues within the Auto-ID Centre labs, Sun Microsystems and the PML Software Action Group.

9. REFERENCES

1. **S. Sarma, D. L. Brock & K. Ashton, "The Networked Physical World – Proposals for Engineering The Next Generation of Computing, Commerce & Automatic Identification".**
October 2000, http://www.autoidcenter.org/research/MIT-AUTOID-WH-001.pdf

2. **D. L. Brock, "Electronic Product Code™ (EPC™) – A Naming Scheme for Physical Objects".**
January 2001, http://www.autoidcenter.org/research/MIT-AUTOID-WH-002.pdf

3. **S. Sarma, "Towards the 5¢ Tag".**
November 2001, http://www.autoidcenter.org/research/MIT-AUTOID-WH-006.pdf

4 **Oat Systems & MIT Auto-ID Center, "The Savant™ – Version 0.1 (Alpha)".**
February 2002, http://www.autoidcenter.org/research/MIT-AUTOID-TM-003.pdf

5 **Oat Systems & MIT Auto-ID Center, "The Object Name Service (ONS) – Version 0.5 (Beta)".**
February 2002, http://www.autoidcenter.org/research/MIT-AUTOID-TM-004.pdf

6. **M. Harrison & D. McFarlane, "Development of a Prototype PML Server for an Auto-ID Enabled Robotic Manufacturing Environment".**
February 2003, http://www.autoidcenter.org/research/CAM-AUTOID-WH010.pdf

7. **Domain Name Service**
for more information see links at http://www.dns.net/dnsrd/

8. **D. L. Brock, "The Physical Markup Language (PML) – A Universal Language for Physical Objects".**
February 2001, http://www.autoidcenter.org/research/MIT-AUTOID-WH-003.pdf

9. **D. L. Brock, T. P. Milne, Y. Y. Kang & B. Lewis, "The Physical Markup Language".**
June 2001, http://www.autoidcenter.org/research/MIT-AUTOID-WH-005.pdf

10. **XML Schema**
http://www.w3.org/XML/Schema

11. **XML – extensible Markup Language**
http://www.w3.org/XML/

12. **Structured Query Language (SQL)**
see also http://www.sql.org

13. **XPath – XML Path Language**
http://www.w3.org/TR/xpath

14. **XQL – XML Query Language**
http://www.w3.org/TandS/QL/QL98/pp/xql.html

15. **Java beans and accessor methods**
http://java.sun.com/products/javabeans/

16. **For details of XML markup languages for various sectors of industry, see**
http://web.mit.edu/mecheng/pml/standards.htm

17. **Many commercial relational databases are developing translation layers for mapping between XML and relational database queries.**
For further examples of translation layers for interfacing between SQL and XML Schemas, see also: http://developer.netspective.com/xif.html and http://zsqlml.sourceforge.net

18. **Castor data binding project from Exolab**
http://castor.exolab.org

19. **Lightweight Directory Access Protocol (LDAP)**
http://www.openldap.org, http://www.ldapman.org and http://www.rage.net/ldap/links.shtml

20. **Java Architecture for XML Binding (JAXB)**
http://java.sun.com/xml/jaxb/

21. **Java Data Objects (JDO)**
http://java.sun.com/products/jdo/

22. **XML Access Control Language (XACL)**
http://www.trl.ibm.com/projects/xml/xacl/

23. **XML Access Control Markup Language (XACML)**
http://www.oasis-open.org/committees/xacml/

24. **HTML – Hypertext Markup Language**
http://www.w3.org/Markup/

25. **D. L. Brock, "The Virtual Electronic Product Code™".**
February 2002, http://www.autoidcenter.org/research/MIT-AUTOID-WH-011.pdf

Designed by Foxner. www.foxner.com