

# WHITE PAPER

## PML Core Specification 1.0

Christian Floerkemeier, Dipan Anarkat,  
Ted Osinski, Mark Harrison

**AUTO-ID CENTER** UNIVERSITY OF ST. GALLEN, INSTITUTE OF TECHNOLOGY MANAGEMENT, UNTERSTRASSE 22, CH-9000 ST. GALLEN, SWITZERLAND

### ABSTRACT

This report documents the core part of the physical markup language (PML Core). It details the scope of PML Core, its relation to the physical markup language, usage scenarios, requirements, design decisions and XML schemas and sample instance documents.

### STATUS OF THIS DOCUMENT

This section describes the status of this document at the time of its publication. Other documents may supersede this document. The latest status of this document series is maintained at the Auto-ID Center.

This document has been reviewed by Auto-ID Center Members and other interested parties and has been endorsed by the Director of Auto-ID Center. It is a stable document and may be used as reference material or cited as a normative reference from another document.

Comments on this document should be sent to the Auto-ID Software Action Group mailing list [sag-pml@develop.autoidcenter.org](mailto:sag-pml@develop.autoidcenter.org).

# WHITE PAPER

## PML Core Specification 1.0

### Biographies

---



**Christian Floerkemeier**  
Doctoral Candidate, M-Lab ETH Zurich

Christian Floerkemeier is currently a doctoral candidate at the M-Lab ([www.m-lab.ch](http://www.m-lab.ch)), a joint initiative of ETH Zurich and the University of St. Gallen to promote the application of pervasive computing in business environments. He holds a Bachelor and Masters degree in Electrical and Information Science from Cambridge University in the UK. His work at the Auto-ID Center focuses on the development of the Physical Markup Language and the underlying information models.



**Dipan Anarkat**  
EC Systems Analyst  
The Uniform Code Council, Inc.

Dipan Anarkat is an Electronic Commerce Systems Analyst for the Uniform Code Council (UCC). He is responsible for developing global EAN.UCC System standards for identification codes, data carriers, and electronic commerce. In addition to standards development, Mr. Anarkat provides technical and architectural support to the UCC's entities, including UCCnet, RosettaNet and EPCglobal US. Throughout his career in Information Technology, Mr. Anarkat has developed and deployed numerous large-scale enterprise applications and eBusiness solutions using the latest in Internet technologies. Prior to joining the UCC in 2001, Mr. Anarkat was employed by BIT Tech as a Software Engineer. Mr. Anarkat obtained his Bachelor of Science degree in Electronics and Computer Science from the University of Pune, India, and a diploma in Advanced Computing from the University of Pune.

# WHITE PAPER

## PML Core Specification 1.0

### Biographies

---



**Ted Osinski**  
Vice President System Architecture and Strategy, Uniform Code Council, Inc.

Mr. Osinski is the Vice President of System Architecture and Strategy for the Uniform Code Council (UCC). In this role, he is responsible for research and development of the next generation of eCommerce architecture and Radio Frequency Identification (RFID) technologies. He has an extensive and varied background in software development and consulting within the retail, financial services, and government industries. Prior to accepting the position as Vice President, Mr. Osinski was involved in developing and deploying emerging technologies including the first document management and workflow system, Object Oriented Technologies, extensible Markup Language (XML), and distributed Internet applications. He is a Software Action Group (SAG) Overseer and a member of UN/CEFACT and W3C. Mr. Osinski earned his Masters of Business Administration in Information Systems from the Farleigh Dickinson University, New Jersey. He holds a Bachelor of Science degree in Computer Studies from the University of Glamorgan in the United Kingdom.



**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.

# WHITE PAPER

## PML Core Specification 1.0

### Contents

---

1. PML Core Introduction .....	4
1.1. Background Information .....	4
1.2. Document Conventions .....	4
1.3. Scope of this Document .....	4
1.4. PML Objectives and Scope .....	5
1.5. PML Core .....	5
1.6. Document Overview .....	7
1.7. Audience .....	7
2. EPC™ System Network Architecture .....	7
2.1. EPC™ Network Software Architecture Components .....	7
2.2. EPC™ Network Data Standards .....	9
2.3. EPC™ Network Architecture – Across Enterprises .....	9
3. PML Core Requirements.....	10
3.1. Overall Description and Usage Scenarios .....	10
3.2. General Guidelines .....	11
3.3. Data Requirements.....	12
4. PML Core Schema Architecture.....	15
4.1. PML Design Methodology Overview .....	15
4.2. PML Namespace design guidelines .....	16
4.3. PML Core File Structure .....	17
5. PML Core Specification Elements .....	18
5.1. Overview .....	18
6. Appenix .....	28
6.1. XML Schemas .....	28
6.2. XML Instance Files .....	33
6.3. Identifier Representation within PML Core.....	37
7. References .....	39

## 1. PML CORE INTRODUCTION

### 1.1. Background Information

This document draws from the previous work at the Auto-ID Center, and we recognize the contribution of the following individuals: David Brock, Dan Engels, Robin Koh, Tim Milne, Christian Floerkemeier, Brendon Lewis, Yun Kang.

The following papers capture the contributions of these individuals:

- David L. Brock, Timothy P. Milne, Yun Y. Kang & Brendon Lewis, “The Physical Markup Language”. 2001. (See <http://www.autoidcenter.org/publishedresearch/MIT-AUTOID-WH-005.pdf>)
- David L. Brock, “The Physical Markup Language – A Universal Language for Physical Objects”.

### 1.2. Document Conventions

The keywords **MUST**, **MUST NOT**, **REQUIRED**, **SHALL**, **SHALL NOT**, **SHOULD**, **SHOULD NOT**, **RECOMMENDED**, **MAY** and **OPTIONAL**, when they appear in this document, are to be interpreted as described in [RFC 2119] as quoted here:

- **MUST:** This word, or the terms “**REQUIRED**” or “**SHALL**”, means that the definition is an absolute requirement of the specification.
- **MUST NOT:** This phrase, or the phrase “**SHALL NOT**”, means that the definition is an absolute prohibition of the specification.
- **SHOULD:** This word, or the adjective “**RECOMMENDED**”, means that there may exist valid reasons in particular circumstances to ignore a particular item, but the full implications must be understood and carefully weighed before choosing a different course.
- **SHOULD NOT:** This phrase, or the phrase “**NOT RECOMMENDED**”, means that there may exist valid reasons in particular circumstances when the particular behavior is acceptable or even useful, but the full implications should be understood and the case carefully weighed before implementing any behavior described with this label.

**MAY:** This word, or the adjective “**OPTIONAL**”, means that an item is truly optional. One vendor may choose to include the item because a particular marketplace requires it or because the vendor feels that it enhances the product while another vendor may omit the same item. An implementation, which does not include a particular option, **MUST** be prepared to interoperate with another implementation, which does include the option, though perhaps with reduced functionality. In the same vein an implementation, which does include a particular option, **MUST** be prepared to interoperate with another implementation, which does not include the option (except, of course, for the feature the option provides).

### 1.3. Scope of this Document

This report documents the core part of the physical markup language (PML Core). It details the scope of PML Core, its relation to the physical markup language, usage scenarios, requirements, design decisions and XML schemas and sample instance documents.

## 1.4. PML Objectives and Scope

The goal of the “Physical Markup Language” (PML) is to provide a collection of common, standardized vocabularies to represent and distribute information related to EPC™ Network enabled objects.

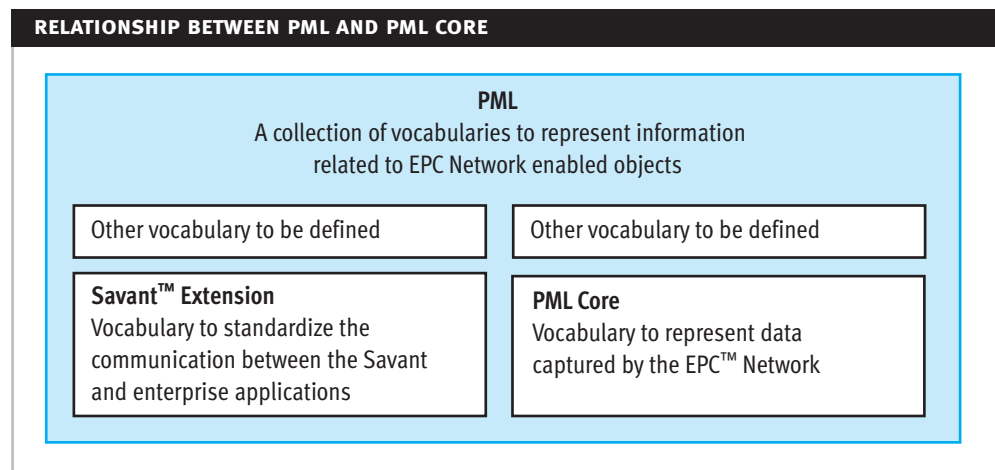
Examples of the kind of content that might be included are observations by sensors such as RFID reads, configuration files for infrastructure components such as RFID readers or e-commerce documents featuring EPC™ data such as advanced shipping notices containing EPCs™ of the items shipped (see Figure 1). Although these different vocabularies might have diverse contents, they will be using naming and design rules common to the PML.

The PML vocabularies provide the XML definitions of the data exchanged between components in the EPC™ Network system. XML messages interchanged in the systems should be instantiated from these PML schemas.

The PML development is part of the Auto-ID Center’s effort to develop standardized interfaces and protocols for the communication with and within the Auto-ID infrastructure.

PML does not attempt to replace existing vocabularies for business transactions or any other XML application libraries, but complements these by defining a new library containing definitions about EPC™ Network system related data.

Figure 1



## 1.5. PML Core

### 1.5.1. Objectives and Scope

The purpose of the core part of the physical markup-language (PML Core) is to provide a standardized format for the exchange of the data captured by the sensors in an Auto-ID infrastructure, e.g. RFID readers.

PML Core provides a set of schemas that define the interchange format for the transmission of the data values captured. These data entities might be accessed directly from the sensor, or from data routers and data stores such as the Savant™ or the EPC™ Information Service that distribute the captured data.

PML Core focuses on observables - physical properties and entities that are capable of being observed or measured by a sensor - rather than the observational and performance characteristics of the individual sensors or the interpretation of the observed values.

Any possible interpretation of these raw data is handled by other vocabularies under the PML umbrella. PML Core is one of the vocabularies in the collection of vocabularies under the PML umbrella (see Figure 1).

### 1.5.2. Motivation

It is believed that by focusing on what is unique to Auto-ID we can provide a vocabulary that suits the needs of the Auto-ID community and at the same time avoids reinventing the wheel by defining a new vocabulary for elements that are already defined in existing business standards such as UBL, EAN.UCC, RosettaNet® and many more (see Technical Memo: Physical Mark-Up Language Update by Christian Floerkemeier and Robin Koh MIT-AUTOID-TM-006 for more details). PML Core hence focuses on providing a flexible framework to represent data captured by sensors in the EPC™ Network.

### 1.5.3. Usage

Messages based on the PML Core schema can be exchanged between any two XML enabled systems in the EPC™ Network. Typically the information exchange based on the PML Core schema will occur between Savant™ and the EPC™ Information Service and/or other enterprise applications. This does not preclude other opportunities for usage of the PML Core schema. Any other industry vertical or organization with requirements that match the PML Core model may make use of it by importing the schema into their specific XML schema or application. Tool support based on the PML schema may be another such opportunity. In general we can say that PML Core messaging can be accomplished between any 2 systems capable of XML messaging.

Figure 2

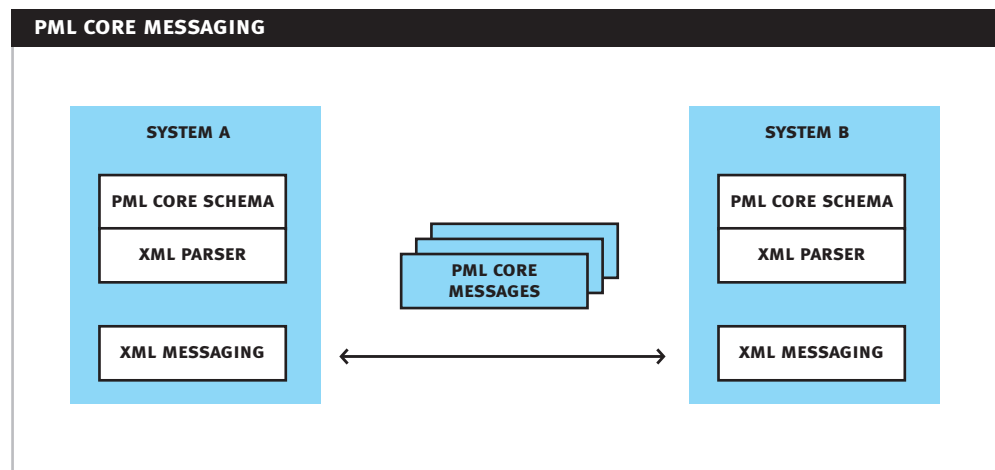
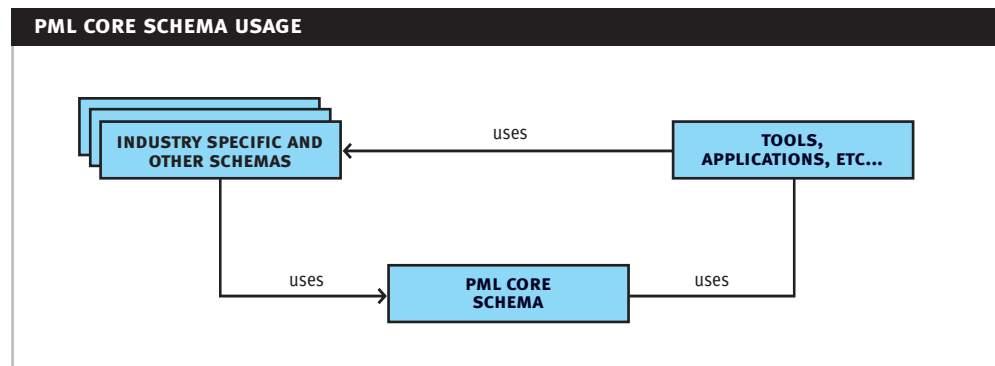


Figure 3



## 1.6. Document Overview

Section 2 of this document contains a general introduction to the various components of the EPC™ Network and how they relate to PML Core. Section 3 lists the features of PML Core needed to fulfill the requirements and to adequately represent the data captured by the EPC™ Network. This section contains general guidelines as well as specific data features required. Section 4 introduces the PML schema architecture and in Section 5 details about the sensor model and the specification of the various elements are provided. The Appendix is comprised of the XML Schemas and XML instances.

## 1.7. Audience

Technical managers and developers are considered to be the primary audience of this document.

# 2. EPC™ SYSTEM NETWORK ARCHITECTURE

Radio Frequency Identification is a technology used to identify, track and locate assets. The vision that drives the developments at the Auto-ID Center is the universal unique identification of individual items. The unique number, called EPC™ (Electronic Product Code™) will be encoded in an inexpensive Radio Frequency Identification (RFID) tag. The EPC™ Network will also capture and make available (via Internet and for authorized requests) other information that pertains to a given item to authorized requestors.

## 2.1. EPC™ Network Software Architecture Components

The EPC™ Network Architecture as in Fig. 4 shows the high-level components of the EPC™ Network.

**Figure 4:** EPC Network Architecture- inside the Enterprise

### ONS (Cache):

- Local copy of frequently used ONS data
- Registration for static & dynamic ONS
- Collaboration on asset tracking

### EPC Information Service:

- Track and trace serial items
- Referencing business transactions
- Object type data (e.g. pallet, case, item, ...)
- Instance level EPC data (e.g. expiry date, ...)
- Fine grained access control policy implementation

### Savant:

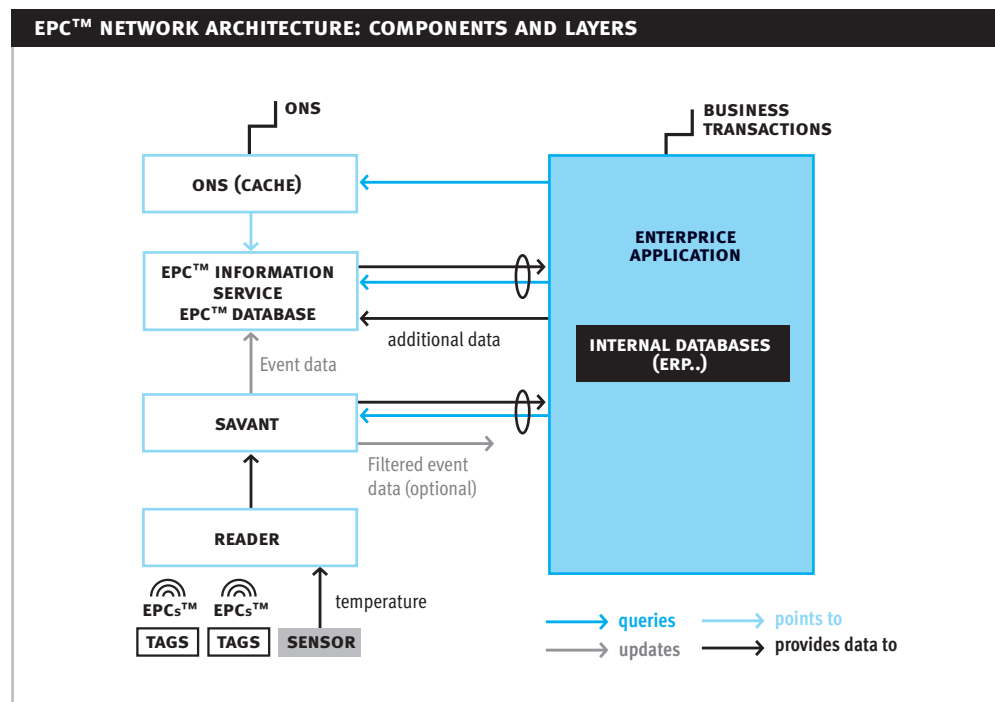
- Report data
- Manage readers
- Higher level filters

### Readers:

- Capture events data (tag & sensors)
- Simple filters

### Tag & Sensor:

- Transmit EPC data using radio frequency
- Transmit sensor data



These functional components from the figures above are described in the sections below.

#### **2.1.1. Readers**

Readers are devices responsible for detecting when tags enter their read range. They may also be capable of interrogating other sensors coupled to tags or embedded within tags.

The Auto-ID Reader Protocol Specification 1.0 defines a standard protocol by which Readers communicate with Savants™ and other hosts. The Savant™ also has an “adapter” provision to interface to older readers that do not implement the Auto-ID Reader Protocol.

#### **2.1.2. Savant™**

Savant™ is “middleware” software designed to process the streams of tag or sensor data (event data) coming from one or more reader devices. Savant™ performs filtering, aggregation, and counting of tag data, reducing the volume of data prior to sending to Enterprise Applications. The Auto-ID Savant™ Specification 1.0 defines the working of Savant™, and the interface to Enterprise Applications.

#### **2.1.3. EPC™ Information Service**

The EPC™ Information Service makes EPC™ Network related data available in PML format to requesting services. Data available through the EPC™ Information Service may include tag read data collected from Savant™ (for example, to assist with object tracking and tracing at serial number granularity); instance-level data such as date of manufacture, expiry date, and so on; and object class-level data such as product catalog information. In responding to requests, the EPC™ Information Service draws upon a variety of data sources that exist within an enterprise, translating that data into PML format. When the EPC™ data is distributed across the supply chain, an industry may create an EPC™ Access Registry that will act as a repository for EPC™ Information Service interface descriptions. The Auto-ID EPC™ Information Service Specification 1.0 defines the protocol for accessing the EPC™ Information Service.

#### **2.1.4. ONS – Object Name Service**

The Object Name Service provides a global lookup service to translate an EPC™ into one or more Internet Uniform Reference Locators (URLs) where further information on the object may be found. These URLs often identify an EPC™ Information Service, though ONS may also be used to associate EPCs™ with web sites and other Internet resources relevant to an object.

ONS provides both static and dynamic services. Static ONS typically provides URLs for information maintained by an object’s manufacturer. Dynamic ONS services record a sequence of custodians as an object moves through a supply chain.

ONS is built using the same technology as DNS, the Domain Name Service of the Internet. The Auto-ID Object Name Service Specification 1.0 defines the working of ONS and its interface to applications.

#### **2.1.5. ONS Local Cache**

The local ONS cache is used to reduce the need to query the global Object Name Service for each object which is seen, since frequently-asked / recently-asked values can be stored in the local cache, which acts as the first port of call for ONS type queries. The local cache may also manage lookup of private internal EPCs™ for asset tracking. Coupled with the local cache will be registration functions for registering EPCs™ with the global ONS system and with a dynamic ONS system for private tracking and collaboration within the supply chain seen by each unique object.

## 2.2. EPC™ Network Data Standards

The operation of the components of the EPC™ Network is subject to data standards that specify the syntax and semantics of data exchanged among components.

### 2.2.1. Electronic Product Code™ (EPC™)

The Electronic Product Code™ is the fundamental identifier for a physical object. The Auto-ID Electronic Product Code™ Data Specification 1.0 defines the abstract content of the Electronic Product Code™, and its concrete realization in the form of RFID tags, Internet URIs, and other representations.

### 2.2.2. Physical Markup Language (PML)

The Physical Mark-Up Language (PML) is a collection of common, standardized XML vocabularies to represent and distribute information related to EPC™ Network enabled objects. The PML standardizes the content of messages exchanged within the EPC™ network. It is, therefore, part of the Auto-ID Center's effort to develop standardized interfaces and protocols for the communication with and within the Auto-ID infrastructure. The core part of the physical mark-up-language (PML Core) provides a standardized format for the exchange of the data captured by the sensors in the Auto-ID infrastructure, e.g. RFID readers. The Auto-ID PML Core specification 1.0 defines the syntax and semantics of PML Core.

## 2.3. EPC™ Network Architecture – Across Enterprises

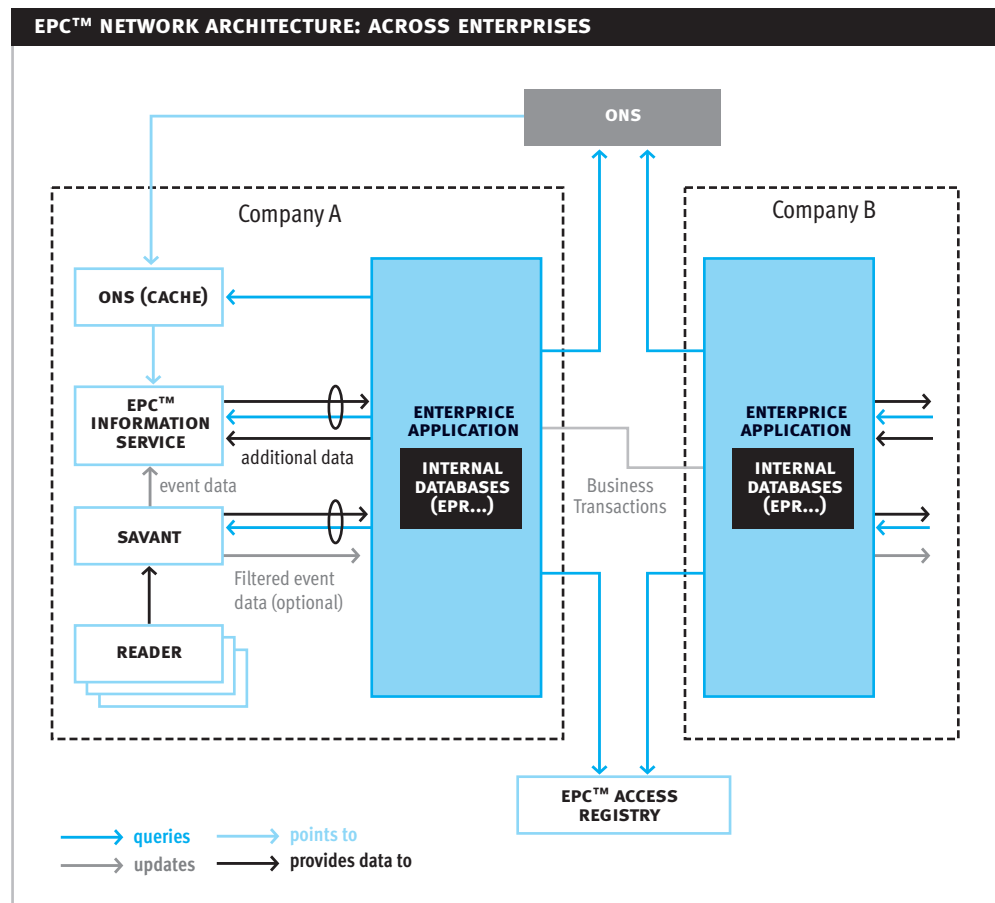
Figure 5:

#### Static ONS

- Converts an EPC™ into an internet address to locate EPC™ Information Service

#### Dynamic ONS

- Provides means to locate current and previous EPC™ custodians for the purpose of track and trace, recall, etc
- Web service interface describing the capabilities and data accessible through each EPC™ information service to trading partners.



### 3. PML CORE REQUIREMENTS

The purpose of this section is to collect, analyze and define the high-level needs and features of PML Core. This requirement section focuses on the capabilities needed by stakeholders and target users and why these needs exist. The details of how the core physical mark-up language fulfills these needs and the design decisions made are detailed in the later sections of this document.

#### 3.1. Overall Description and Usage Scenarios

The PML Core vocabulary should provide the payload mark-up of sensor data communication between:

- A Savant™/EPC™ Information Service and an external application
- Savants™ attached to individual sensors and Savants™ that aggregate information
- A sensor (such as an RFID reader) and a Savant™,

if the processing capabilities of the components allow for an XML based information interchange.

It is supposed to be used as the format for any interchange of the captured data with and within the Auto-ID infrastructure, while being agnostic about how the data are stored or transported.

The following sections outline how PML Core can be used in conjunction with the other EPC™ Network components explained in the previous section:

##### 3.1.1. RFID Readers and Other AIDC Technologies (Such as Bar Code Readers)

The RFID readers and other AIDC technologies detect and identify objects and as such generate the EPC™ data. RFID readers can use PML Core to describe this content with a standard vocabulary for the distribution of the captured data.

##### 3.1.2. Savant™

The Savant™ is the “middleware” of the Auto-ID technology responsible for data processing, routing and filtering. It can make use of the PML Core vocabulary to mark-up captured data it has received from EPC™ Network sensors before the data are distributed to other entities using the transfer routing protocol of choice.

##### 3.1.3. EPC™ Information Service

The EPC™ Information Service is the “point of query” for external applications that need to query EPC™ Network related data. If the queries relate to data captured by the EPC™ network (e.g. RFID reads) the response should be marked up using the PML Core vocabulary.

##### 3.1.4. External Custom Applications

PML Core language provides the common syntax for the data captured by the EPC™ Network and received by these applications.

The physical markup language standard makes no recommendations on how the data exchanged are stored in the various components. A Savant™ or the EPC™ information service for example does not necessarily need to store or process data in PML Core format, since PML Core should only be used to mark-up sensor data, when they are exchanged with other nodes in the EPC™ network.

## 3.2. General Guidelines

### 3.2.1. Use of Existing Standards

DESCRIPTION: Using existing standards to describe and uniquely define individual entities such as date, time is recommended whenever applicable. The requirement to use existing standards also applies to the appropriate choice of naming and design rules and to the choice of a particular schema architecture.

RATIONALE: Alignment with existing standards will ensure speed of development, maximum interoperability and ease of long-term maintenance. It also ensures that the focus and scope of the PML Core development does not creep to include features that are not unique to Auto-ID and are already covered by other efforts.

### 3.2.2. Rigidity

DESCRIPTION: The language should be described in a way that the document structure and the content are constrained.

RATIONALE: Using a rigidly specified language allows for the use of parsers that check the validity of the document and its contents. This should prevent the sort of problems seen with HTML, which did not encourage strict adherence to the syntax, leaving it up to the browser author to decide which violations of syntax to accept.

### 3.2.3. Simplicity

DESCRIPTION: Simplicity means that the use and implementation of the language is straightforward.

RATIONALE: To encourage the adoption of the PML Core language and the Auto-ID technology the PML language should be as simple and expressive as possible.

### 3.2.4. No Assumption About Underlying Transport Protocol

DESCRIPTION: During the design and implementation no specific transport protocol that carries the data from one node to another should be assumed.

RATIONALE: Making no assumption about the underlying transport protocol allows us to pick an appropriate transport protocol at a later stage without being constraint by our initial choice for the design and implementation of PML Core.

### 3.2.5. Human Readability

DESCRIPTION: By human readability it is meant that the semantics of data fields are not made less obvious by choosing cryptic names.

RATIONALE: The rationale for human readability is that it increases the learning curve and simplifies the debugging process. It is common practice in today's XML standards development. The disadvantage of human readability and expressive names is that more data have to be transferred. It is however believed that these bandwidth savings are not large enough to justify the use of cryptic tag names.

### 3.2.6. Availability of Tools for Schema Validation Language and Authoring

DESCRIPTION: To support the use of PML Core the user will need to rely on tools to author files in the specified syntax and validate its content against the schema of the language.

RATIONALE: Without support tools the adoption of PML is endangered because of lacking vendor support.

### **3.2.7. Enable Maximum Component Reuse**

DESCRIPTION: The language should be designed in such a way that the individual components can be reused in different context settings.

RATIONALE: Designing with component reuse in mind the PML Core building blocks can be reused in a context that might not be envisioned during the initial design.

### **3.2.8. 80/20 Rule**

DESCRIPTION: The design of PML Core should provide 20% of the features that accommodate 80% of the needs.

RATIONALE: As mentioned above PML Core should have a simple design. By including special needs that only a very few users require, the vocabulary would become rather complex and difficult to use.

### **3.2.9. Tool Use and Support**

DESCRIPTION: PML Core should make no assumption about tools for creation, management, storage or presentation being available.

RATIONALE: To ease adoption of the PML Core vocabulary we should not restrict its usage by relying on specific tools to create, manage or store the data.

### **3.2.10. Interchange Use**

DESCRIPTION: PML Core is intended for interchange and application use. It should not make any assumptions or recommendations about how the data are actually stored.

RATIONALE: The goal of PML Core is to standardize the mark-up of data captured by the EPC™ Network. By assuming certain storage mechanism, e.g. XML databases, we would unnecessarily endanger the adoption since parties implementing PML Core would be forced to adopt those storage recommendations as well.

## **3.3. Data Requirements**

The following section outlines the data types that were believed to be required in PML Core.

### **3.3.1. Data Captured by RFID Readers**

DESCRIPTION: RFID readers capture the Electronic Product Code™ (in various representations) stored on the individual Auto-ID compliant tags. PML Core should be able to represent this sensing process, where a certain RFID reader, which is identified by a unique identifier, observes/detects certain tags in its read range at a certain moment in time. Each such observation might need to be labeled with the command that was issued to trigger the observation and a unique label to reference a certain observation.

RATIONALE: RFID readers are one of the main components within the EPC™ Network. The data they capture are routed within the EPC™ Network from readers to Savant™, from one Savant™ to other, from Savant™ to the EPC™ Information Service. To standardize the mark-up of those captured data, PML Core needs to adequately represent the observed values.

The unique label is needed to reference certain observations once they are used to infer certain high-level information e.g. certain RFID reads at a dock door are interpreted as the arrival of a shipment. To reference the cause of this interpretation, it might be useful to reference the actual observation.

The command that was issued e.g. to make the RFID reader scan its read range, is needed because the reader itself might support a variety of measurement modes. To interpret the observed value accordingly the command issued will help to provide further insights.

PRIORITY: Must have

### **3.3.2. Data Captured by Non-RFID Identification Sensors**

DESCRIPTION: Non-RFID identification sensors such as bar code scanners capture similar information when compared to RFID identification sensors. The actual data requirements are hence also similar to the ones mentioned for RFID readers.

RATIONALE: To ease adoption existing identification systems such as bar code scanners should be supported.

PRIORITY: Should have

### **3.3.3 Data Generated by Sensors Mounted on RFID Tags**

DESCRIPTION: RFID tags may contain sensors, which observe certain environmental phenomena and make the observed values available. The mounted sensors can for example include temperature sensors, humidity sensors or weight sensors. Each sensor observation requires its own timestamp and potentially the command that was issued to make the measurement.

RATIONALE: Future generation of Auto-ID tags will contain active RFID tags that have on-board sensors. To adequately represent the data these sensor capture PML Core needs to be able to model the observed values.

PRIORITY: Should have

### **3.3.4. Data Captured by Fixed Wired Sensors that Monitor Physical Properties**

DESCRIPTION: Fixed wired sensors monitor the environment and provide data such as the temperature at a certain location or the weight of a certain item. Similar to sensors mounted on tags they observe a certain physical property and make the observed value available. This value can be a single data entity, a vector of data entities or an aggregate such an average, maximum or minimum.

The actual data requirements of this item are similar to the one for sensors mounted on RFID tags (see previous requirement). It is nonetheless included to underline that we need to consider data captured by wired and wireless sensors.

RATIONALE: Fixed wired sensors that monitor physical properties augment the data captured by identification sensors such as RFID readers or Bar Code scanners. In order to use this information together with location information provided by the identification sensors, a standardized format to represent the observed physical properties is needed.

PRIORITY: Should have

### **3.3.5. Hierarchy of Sensor Observations**

DESCRIPTION: A hierarchy of sensor observations occurs, when an RFID tag with sensing capabilities is present. The on-board sensors measure a certain physical property, store the observed values and transmit them, once they are in the vicinity of an RFID reader and the RFID tag is detected. Each observation/measurement needs its own timestamp so that the individual observations can be distinguished once they are transmitted from the active tag to the RFID reader.

**RATIONALE:** The hierarchy of sensor observations is a direct consequence of the availability of sensors mounted on RFID tags.

**PRIORITY:** Should have

### **3.3.6. Representation of Generic Sensor Observations**

**DESCRIPTION:** Generic sensor observations do not indicate the semantics of the observed data. Consider a particular RFID reader, which makes the observed data, which might include collisions on the air interface, CRC errors and the EPCs™ of the tags detected in hex notation, available as a byte array. Rather than representing the observed value in an expressive format where it is evident that certain tags are detected and that collisions occurred, the availability of generic sensor observations allows the sensor to report its observation as a data field (“a data blob”), whose semantics can only be accessed by referring to additional documentation provided by the sensor.

**RATIONALE:** The representation of generic sensor observations provides flexibility to PML Core so that it can be used to represent data entities that are e.g. at a fairly low-level and would require a much more extensive PML Core, since all possible observed values would need to be modeled explicitly.

**PRIORITY:** Must have

### **3.3.7. Representation of Sensor Specific Observations**

**DESCRIPTION:** Sensor specific observations distinguish themselves from the generic ones by explicitly saying of what type the observed value is. Rather than representing tag reads as a blob of data, whose meaning can only be retrieved by accessing the documentation provided by the sensor, the observations are represented e.g. as tag reads or temperature values explicitly.

**RATIONALE:** The EPC™ network is mainly believed to capture data from RFID readers and sensors observing the environment such as temperature sensors. To represent these predominant types of observations we need to model them explicitly. Otherwise, application developers are required to implement the conversion from a generic data blob containing tag reads to the individual EPCs™ detected by them.

**PRIORITY:** Must have

### **3.3.8. Openness for Different Kinds of Sensor Observations**

**DESCRIPTION:** The PML Core schemas should enable instance document authors to create instance documents containing elements above and beyond what was specified by the PML Core schemas with respect to observations the sensors make and how the sensors are configured.

**RATIONALE:** This localized openness allows instance document authors to describe features that were not anticipated by the schema designers of PML Core. This is required especially in light of the development of Auto-ID compliant active tags, which will offer many more possibilities with respect to memory structure, mounted sensors and access control. PML Core needs to facilitate these additional features without necessarily specifying at this stage what the various features are.

**PRIORITY:** Should have

### **3.3.9. Represent Tags With and Without Memory**

**Description:** Tags can either store an identifier only or also have additional memory to store random data.

**RATIONALE:** Although the early EPC™ compliant tags will store an identifier only, later generation of Auto-ID Center tags might feature additional memory.

**PRIORITY:** Should have

### **3.3.10. Make the EPC™ the Default Identification Scheme**

**DESCRIPTION:** To uniquely identify sensors, tags and other objects within the EPC™ Network unique identification schemes are required. The EPC™ should be the default identification scheme. Under exceptional circumstances other identification schemes can be used, if the type of identification scheme is properly specified.

**RATIONALE:** The EPC™ is one of the main components of the EPC™ Network and its use should therefore be promoted within PML Core.

**PRIORITY:** Must have

## **4. PML CORE SCHEMA ARCHITECTURE**

As PML Core is a subset of PML, the PML Core schemas follow the PML design methodology. The specifications provide nomenclature, design principles and best practices for PML Core schema development. From a PML Core standpoint it is enough for the reader to know that a standard and well-defined XML design methodology has been applied to produce quality PML Core schemas. Interested readers may further explore the details of the XML design methodology as described below, but is not required to know the same for implementing PML Core schemas.

### **4.1. PML Design Methodology Overview**

PML uses the W3C XML Schema language [XSD] as the schema meta-language for its definition. Although different syntactic representations could be used, XML Schema has been well defined and in general use as a simple method for embedded meta-data in flexible structures.

Any standardized XML vocabulary needs to have a documented and well-defined design methodology for ease of understanding, adoption and implementation. A well-defined XML design methodology documents the design principles used to architect a particular XML Schema vocabulary. A XML design methodology standardizes design principles such as:

- Naming and design rules for schema files and components
- Versioning of schemas and components
- Namespace definition and use
- Complexity, generality and modularity of schemas and components
- Component reusability
- Schema documentation

Rather than reinvent a new XML design methodology of its own, PML makes use of an existing and well-defined methodology for its design. The design of PML is based on the XML design methodology as defined by RosettaNet®. The details of this methodology are beyond the scope of this document. The XML design methodology is defined in the following 3 RosettaNet® specifications:

1. Universal Structures (hereafter [UST])
2. XML Design Guidelines (hereafter [XMLDG])
3. Namespace Specification and Management (hereafter [NSSM])

Further details about [UST], [NSSM] and [XMLDG] can be found in the ‘References’ section of this document. This methodology as adopted and extended for use in PML development is hereafter referred to as PML design methodology. To understand the details of the PML design methodology, the reader should be familiar with the above-mentioned specifications as the design of PML is based on these specifications.

## 4.2. PML Namespace Design Guidelines

Uniform Resource Names (URNs) serve as persistent, location-independent, resource identifiers. This section describes the structure of legal URNs for Auto-ID XML resources as covered by PML. All PML resources MUST adopt the namespace design guidelines as outlined in the section below. These guidelines are based on the [NSSM] specification that provides guidelines for designing hierarchical namespace URNs. Though these guidelines have been established as part of the PML design methodology effort they can be extended and applied to all Auto-ID resources.

### 4.2.1. Namespace Hierarchy

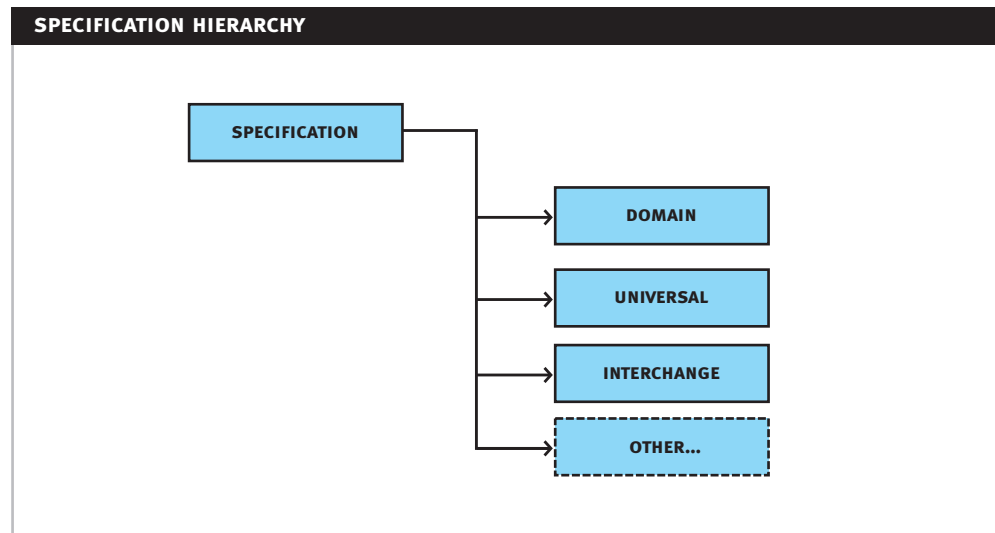
Namespaces are formatted as URNs and have a hierarchical structure. Each hierarchical level of the namespace URN provides additional information about the specification entity being identified by the name space in consideration. The Namespace ID (NID) to be used for all Auto-ID namespaces is “autoid”. For the Namespaces Specific String (NSS) part of a URN we defined the following hierarchical structure [RFC 2141] with one branch “specification” at the top of the hierarchy. In the future Auto-ID may define additional top-level branches as required.

Note that RFC 2141 specifies both “urn” and NID to be case-insensitive, however NSS in Auto-id namespaces is to be considered as case-sensitive.

#### 4.2.1.1. “SPECIFICATION” HIERARCHY

The “specification” hierarchy consists of published Auto-ID specifications. As described in the Figure 5 below, the specifications can belong to many specification classes, such as domain, universal, interchange, or to an as yet unspecified specification class. The specification may be schemas, text documents, etc. The “specification” hierarchy also requires mandatory versioning of all URNs for reusable or referable resources within this hierarchy.

Figure 6



The “specification” hierarchy is described below:

```
urn:autoid:specification:{ specification-class }:{ specification-
subclass? }:{ specification-id? }:{ type }:{ :subtype? }{ :document-id? }{ :version-id}
```

**specification-class ::= domain|universal|interchange|...**

**specification-class** is the class of a specification. “**domain**” identifies resources that are defined in that particular domain. “**universal**” identifies resources that are universal in the “**autoid**” namespace. “**interchange**” identifies resources that are interchanged between components in the Auto-ID system, example; XML messages.

**specification-subclass ::= Savant|Reader|...**

**specification-subclass** should be used wherever it is applicable to identify subclasses within **specification-class**.

**specification-id** is a unique identifier within the specification-class for the resource, and MUST be the same as an existing name within the **specification-subclass** (or **specification-class** as the case may be).

**type ::= xml| ...**

**type** is the type of the resource, and MUST be easily understood and recognized by Auto-ID audience. The value of “**type**” MUST be ‘xml’ for all PML resources.

**sub-type::=schema|soap-rpc|stylesheet|service|...**

**sub-type** is an optional sub-type of the resource, and MUST be easily understood and recognized by Auto-ID audience.

**document-id** is an optional identifier of the document to which the resource is related to, and MUST be **version-id ::= {major}**

Table 1: Example values for the field

SPECIFICATION-CLASS	SPECIFICATION-SUBCLASS	SPECIFICATION-ID	TYPE	SUBTYPE	DOCUMENT-ID	VERSION-ID
universal		Identifier	xml	schema		1
interchange		PMLCore	xml	schema		1
interchange	Savant	core	xml	soap-rpc		1
interchange	Savant	core	xml	service		1
interchange	Savant	readerproxy	xml	soap-rpc		1
interchange	Savant	readerproxy	xml	service		1

The URNs constructed with these values are:

- urn:autoid:specification:universal:Identifier:xml:schema:1
- urn:autoid:specification:interchange:PMLCore:xml:schema:1
- urn:autoid:specification:interchange:Savant:core:xml:soap-rpc:1
- urn:autoid:specification:interchange:Savant:core:xml:service:1
- urn:autoid:specification:interchange:Savant:readerproxy:xml:soap-rpc:1
- urn:autoid:specification:interchange:Savant:readerproxy:xml:service:1

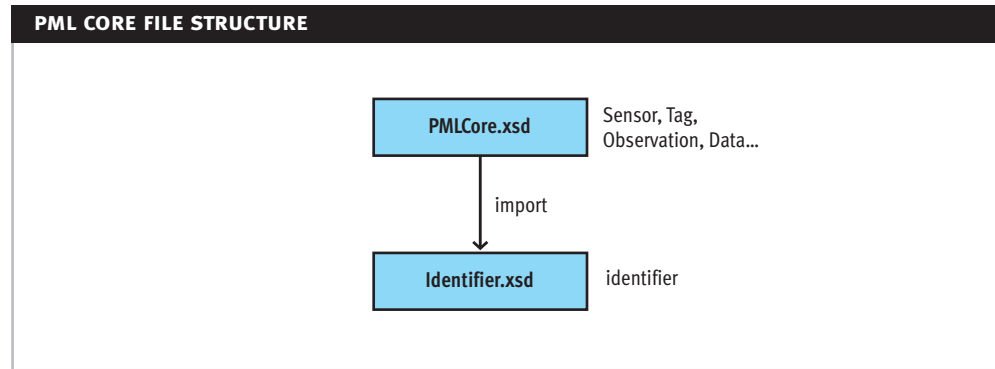
### 4.3. PML Core File Structure

PML Core specification elements as specified in the next section are defined in the ‘PMLCore.xsd’ file. As explained earlier, PML Core defines the sensor data model. ‘Sensor’ is the global element in the PML Core schema based on which XML messages should be instantiated for interchange of sensor

data between two XML enabled components in the EPC™ Network system. 'PMLCore.xsd' follows the methodology guidelines as specified in the sections above. It is an 'interchange' schema as it defines the 'Sensor' interchange element.

'PMLCore.xsd' imports the 'Identifier.xsd' PML schema. 'Identifier.xsd' is a Universal schema as it defines the 'Identifier' structure that is truly universal and not specific to PML Core.

Figure 7



## 5. PML CORE SPECIFICATION ELEMENTS

### 5.1. Overview

The PML Core Sensor Model is comprised of the following components:

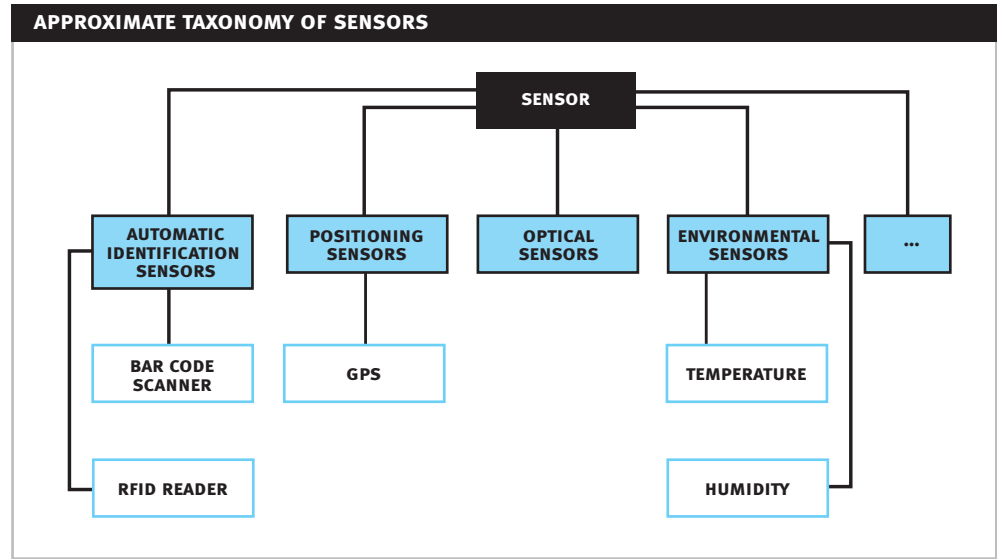
- **Sensors** – Sensors are considered devices that are capable of making measurements of physical properties and entities. Examples include RFID readers, bar code scanners, temperature sensors and weight devices (see Figure 8).
- **Observations** – Observations represent measurements made by the sensors. They associate the actual observed data with the sensor.
- **Observables** – Observables are physical properties and entities that are capable of being observed by the sensors. This includes for example tags detected or temperature and humidity values measured.

PML Core is hence based on a model, in which an observer or sensor makes certain observation of certain observables.

It is worthwhile to stress that RFID readers are considered just another type of sensors when compared to temperature or weight sensors. RFID readers are therefore not explicitly modeled in the PML Core Sensor Model. Just as thermometer, humidity sensors, bar code scanners, GPS devices it is just represented as a sensor.

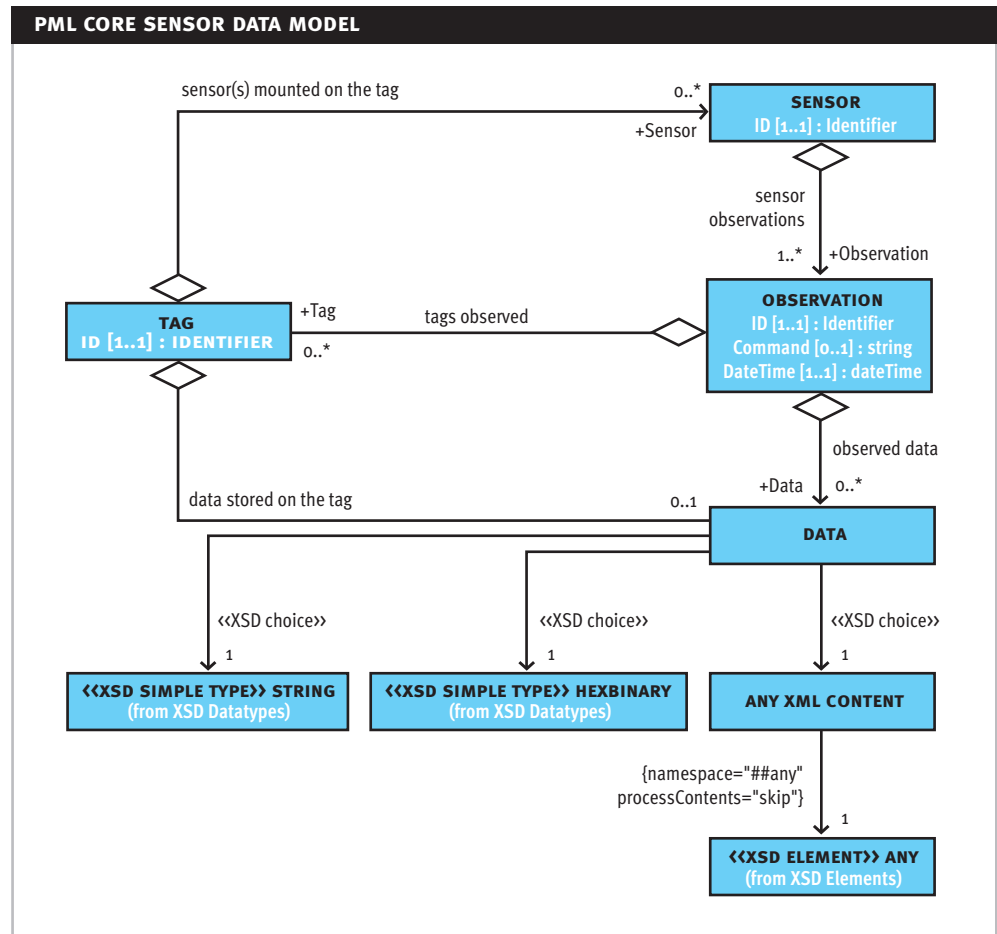
The rationale for representing an RFID reader not explicitly as such is that PML Core should provide a generic flexible framework for sensor observation within the EPC™ Network. By explicitly modeling RFID readers we would constrain this framework to RFID readers, and thus limiting the use of PML Core. Other sensor types would then need an explicit representation as well.

**Figure 8:** Approximate taxonomy of sensors – illustrating that PML Core represents all these different devices as sensors. They are not explicitly represented in PML Core.



A common property to any sensor or tag device in the EPC™ Network is that it has an identification. The ID of the device should by default be an EPC™ but is not limited to the same. Sensors could also have any proprietary form of identification.

**Figure 9**



The PML Core specification elements are defined in 'PMLCore.xsd' XML schema file which can be found in Appendix A of this specification document. The schema is normative and takes precedence over the text contained herein in case of any ambiguity.

#### 5.1.1.1. Sensor Element

The **Sensor** element is the main interchange element for PML Core messaging. This element is a composite element comprised of the following subordinate elements:

- **ID** element
- one or more **Observation** elements (see section 5.1.2 for details)

The **Sensor** element captures sensor information. As mentioned earlier, a sensor is considered any device that makes measurements and observations. In the PML Core sensor model there is hence no distinction between an RFID reader and a temperature sensor except that they have a different identifier code. The different identifier code and the information that can be retrieved using this identifier allow applications to gain further insights into the sensing process. The data entities that can be retrieved this way might include the following items – none of which will be part of PML Core though:

- Quality characteristics (e.g. accuracy)
- Performance characteristic (e.g. sampling rate)
- Orientation and location of the sensor

Requirement Trace:

- Data captured by RFID readers
- Data captured by non-RFID identification sensors
- Data generated by sensors mounted on RFID tags
- Data captured by fixed wired sensors that monitor physical properties

##### 5.1.1.1.1. ID ELEMENT

Sensors in the EPC™ Network are identified by an identifier code. The default identification scheme should be the EPC™. The attributes of the identifier can however be used to specify an alternate identification scheme under exceptional circumstances. The universal structure **ID** element is reused to capture sensor identification information. See section 5.1.5 for details of the **ID** element.

Requirement Trace:

- Make the EPC™ the default identification scheme

##### 5.1.1.2. SAMPLE XML FOR SENSOR ELEMENT

```
<pmlcore: Sensor>
  <pmluid:ID>urn:epc:1:4.16.36</pmluid:ID>
  <pmlcore:Observation>
    <pmlcore:DateTime>2002-11-06T13:04:34-06:00</pmlcore:DateTime>
    <pmlcore:Tag>
      <pmluid:ID>urn:epc:1:2.24.400</pmluid:ID>
    </pmlcore:Tag>
    <pmlcore:Tag>
      <pmluid:ID>urn:epc:1:2.24.401</pmluid:ID>
    </pmlcore:Tag>
  </pmlcore:Observation>
</pmlcore:Sensor>
```

### 5.1.2. Observation Element

Each **Observation** element contains data that are the result of a measurement by a particular sensor. Each observation must be labeled with date and time. It can also be equipped with a unique ID, and a reference to the kind of command that was issued to make the observation.

The **Observation** element consists of the following:

- an optional **ID** element
- an optional **Command** element
- **DateTime** element
- zero or more **Data** elements (see section 5.1.3 for details)
- zero or more **Tag** elements (see section 5.1.4 for details)

#### 5.1.2.1. DATETIME ELEMENT

The **DateTime** element captures the date and time when the observation was made. It is based on the [XSD] data type 'dateTime'.

#### 5.1.2.2. ID ELEMENT

PML Core does not define the actual format of the unique identifier of the individual observation. It is an optional field that allows developers to label the observation uniquely. The universal structure **ID** element is reused to capture unique identification information about the observation made. See section 5.1.5 for details of the **ID** element.

Requirement Trace:

- Data captured by RFID readers

#### 5.1.2.3. COMMAND ELEMENT

The **Command** element can be used to specify the command that was issued to trigger the observation. For an RFID reader this could for example be "read pallet tags only". The various command sets that are available for a certain sensor are detailed in the documentation of the sensor (outside the scope of PML Core).

Requirement Trace:

- Data captured by RFID readers

#### 5.1.2.4. SAMPLE XML FOR OBSERVATION ELEMENT

```
<pmlcore: Sensor>
  <pmluid:ID>urn:epc:1:4.16.36</pmluid:ID>
  <pmlcore:Observation>
    <pmluid:ID>00000001</pmluid:ID>

    <pmlcore:DateTime>2002-11-06T13:04:34-06:00</pmlcore:DateTime>
    <pmlcore:Command>READ_PALLET_TAGS_ONLY</pmlcore:Command>
    <pmlcore:Tag>
      <pmluid:ID>urn:epc:1:2.24.400</pmluid:ID>
    </pmlcore:Tag>
    <pmlcore:Tag>
      <pmluid:ID>urn:epc:1:2.24.401</pmluid:ID>
    </pmlcore:Tag>
  </pmlcore:Observation>
</pmlcore:Sensor>
```

### 5.1.3. Data Element

The **Data** element must be used to represent the data captured when a sensor measured a particular property or entity, unless the data captured can be represented as **Tag** elements (see **Tag** element specification)

It can be used to either represent unstructured data as a simple data blob or structured data by inserting additional XML tags from a different namespace. The schemas for those XML instances are not within the scope of this PML Core version.

The **Data** element consists of a choice of the following 3 elements:

- **Text** element
- **Binary** element
- **XML** element

#### 5.1.3.1. TEXT ELEMENT

The **Text** element must be used to represent captured data as a “data blob” in string notation. It uses the [XSD] ‘string’ data type.

The **Text** element reflects the requirement to represent generic sensor observations, since the data are represented as a data blob and its semantics are not indicated. Tag reads by an RFID reader can for example be represented in this field as a byte array featuring collisions on the air interface or CRC errors. The semantics of this byte array would be documented in the information provided by the sensor. The unique ID of the sensor can be used as a reference to access this information. Another example would be a set of numbers representing temperature readings by a temperature sensor.

Requirement Trace:

- Openness for different kinds of sensor observations
- Representation of generic sensor observations

#### 5.1.3.2. SAMPLE XML FOR TEXT ELEMENT

```
<pmlcore:Sensor>
  <pmluid:ID>urn:epc:1:124.162.37</pmluid:ID>
  <pmlcore:Observation>
    <pmlcore:DateTime>2002-11-06T13:04:34-06:00</pmlcore:DateTime>
    <pmlcore:Data>
      <pmlcore:Text>temp=22,24,25,22,22,23,22</pmlcore:Text>
    </pmlcore:Data>
  </pmlcore:Observation>
</pmlcore:Sensor>
```

#### 5.1.3.3. BINARY ELEMENT

The **Binary** element must be used to represent captured data as a data blob in ‘hexbinary’ notation. It uses the [XSD] ‘hexbinary’ data type.

Similar to the **Text** element, the **Binary** element reflects the requirement to represent generic sensor observations, since the data are represented as a data blob and its semantics are not indicated.

Requirement Trace:

- Openness for different kinds of sensor observations
- Representation of generic sensor observations

#### 5.1.3.4. SAMPLE XML FOR BINARY ELEMENT

```
<pmlcore:Sensor>
  <pmluid:ID>urn:epc:1:124.162.37</pmluid:ID>
  <pmlcore:Observation>
    <pmlcore:DateTime>2002-11-06T13:04:34-06:00</pmlcore:DateTime>
    <pmlcore:Data>
      <pmlcore:Binary> 0FB8A0F5CB0F11000FB8A0F5CB0F11000FB8A0F5CB0F1100</
        pmlcore:Binary>
      </pmlcore:Data>
    </pmlcore:Observation>
  </pmlcore:Sensor>
```

#### 5.1.3.5. XML ELEMENT

The **XML Element** must be used when the instance author wants to represent captured data with XML elements that go beyond what is specified in PML Core.

The [XSD] ‘any’ element enables instance document authors to create instance documents containing elements above and beyond what is specified by the PML Core schema. The instance documents are hence extensible. This should be contrasted with the remainder of the PML Core schema where the content of all elements is always fixed and static. By inserting this localized openness we are empowering the instance document author with the ability to define what data makes sense to him/her. The rationale for the localized openness is that we have to design the PML Core schemas with the recognition that, as schema designers, we can never anticipate all the different kinds of data instance document authors will want to use in the instance document when they represent measurements by various authors.

#### Requirement Trace:

- Openness for different kinds of sensor observations
- Data generated by sensors mounted on RFID tags
- Data captured by fixed wired sensors that monitor physical properties
- Representation of sensor specific observations

#### 5.1.3.6. SAMPLE XML FOR XML ELEMENT

```
<pmlcore:Sensor>
  <pmluid:ID>urn:epc:1:124.162.37</pmluid:ID>
  <pmlcore:Observation>
    <pmlcore:DateTime>2002-11-06T13:04:34-06:00</pmlcore:DateTime>
    <pmlcore:Data>
      <pmlcore:XML>
        <TemperatureReading xmlns="http://sensor.example.org/">
          <Unit>Celsius</Unit>
          <Value>5.3</Value>
        </TemperatureReading>
      </pmlcore:XML>
    </pmlcore:Data>
  </pmlcore:Observation>
</pmlcore:Sensor>
```

#### 5.1.4. Tag Element

The tag element is a special kind of observed value introduced with recognition of the importance of automatic identifications in the EPC™ network. The “tag” entity represents any device that can be detected by a sensor. It may contain memory to store random data and it may itself contain other sensor e.g. a temperature sensor. It does however not necessarily need to be an electronic device such as an RFID tag, since a bar code detected by a bar code scanner would also be considered a “tag”. The tag entity is defined by the **Tag** element.

The **Tag** element consists of the following elements

- **ID** element
- optional **Data** element
- zero or more **Sensor** elements

Requirement Trace:

- Data captured by RFID readers
- Data captured by non-RFID identification sensors
- Representation of sensor specific observations

##### 5.1.4.1. ID ELEMENT

Tags in the EPC™ Network are identified by an identifier code. The universal structure **ID** element is reused to capture tag identification information. See section 5.1.5 for details of the **ID** element. The default identification scheme should be the EPC™. The attributes of the **ID** element can however be used to specify an alternate identification scheme under exceptional circumstances.

Requirement Trace:

- Make the EPC™ the default identification scheme

##### 5.1.4.2. SAMPLE XML FOR TAG ELEMENT

```
<pmlcore: Sensor>
  <pmluid:ID>urn:epc:1:4.16.36</pmluid:ID>
  <pmlcore:Observation>
    <pmlcore:DateTime>2002-11-06T13:04:34-06:00</pmlcore:DateTime>
    <pmlcore:Tag>
      <pmluid:ID>urn:epc:1:2.24.400</pmluid:ID>
    </pmlcore:Tag>
    <pmlcore:Tag>
      <pmluid:ID>urn:epc:1:2.24.401</pmluid:ID>
    </pmlcore:Tag>
  </pmlcore:Observation>
</pmlcore:Sensor>
```

##### 5.1.4.3. DATA ELEMENT

The tag **Data** element is used to make random data available that was stored on a tag if it provides this feature. See Section 5.1.3 for more details about the **Data** element. It allows for the representation of data blobs as well as structured data in the form of XML instances under a different namespace.

Requirement Trace:

- Represent tags with and without memory

#### 5.1.4.4. SAMPLE XML DATA ELEMENT

```
<pmlcore: Sensor>
  <pmluid:ID>urn:epc:1:4.16.36</pmluid:ID>
  <pmlcore:Observation>
    <pmlcore:DateTime>2002-11-06T13:04:34-06:00</pmlcore:DateTime>
    <pmlcore:Tag>
      <pmluid:ID>urn:epc:1:2.24.400</pmluid:ID>
      <pmlcore:Data>
        <pmlcore:XML>
          <EEPROM xmlns="http://sensor.example.org/">
            <FamilyCode>12</FamilyCode>
            <ApplicationIdentifier>123<
              /ApplicationIdentifier>
            <Block1>FFA0456F</Block1>
            <Block2>00000000</Block2>
          </EEPROM>
        </pmlcore:XML>
      </pmlcore:Data>
    </pmlcore:Tag>
  </pmlcore:Observation>
</pmlcore:Sensor>
```

#### 5.1.4.5. SENSOR ELEMENT

Sensors mounted on a tag can make observations independent of the sensor detecting the tag that contains the sensors. The observations made by these sensors are represented as detailed in the above sections on sensor observations. The **Sensor** element contained in the **Tag** element is used provide for the information captured by sensors on mounted on tags. It reflects a recursive structure, where a sensor detects a tag that contains other sensors.

#### Requirement Trace:

- Hierarchy of sensor observations
- Data generated by sensors mounted on RFID tags

#### 5.1.4.6. XML SAMPLE FOR SENSOR ELEMENT

```
<pmlcore:Sensor>
  <pmluid:ID>urn:epc:1:4.16.36</pmluid:ID>
  <pmlcore:Observation>
    <pmluid:ID>00000001</pmluid:ID>
    <pmlcore:DateTime>2002-11-06T13:04:34-06:00</pmlcore:DateTime>
    <pmlcore:Tag>
      <pmluid:ID>urn:epc:1:2.24.400</pmluid:ID>
      <pmlcore:Sensor>
        <pmluid:ID>urn:epc:1:12.8.128</pmluid:ID>
        <pmlcore:Observation>
          <pmlcore:DateTime>2002-11-06T11:00:00-06:00</pmlcore:DateTime>
          <pmlcore:Data>
            <pmlcore:XML>
              <TemperatureReading xmlns="http://sensor.example.org/">
                <Unit>Celsius</Unit>
              </TemperatureReading>
            </pmlcore:XML>
          </pmlcore:Data>
        </pmlcore:Observation>
      </pmlcore:Sensor>
    </pmlcore:Tag>
  </pmlcore:Observation>
</pmlcore:Sensor>
```

```
        <Value>5.3</Value>
      </TemperatureReading>
    </pmlcore:XML>
  </pmlcore:Data>
</pmlcore:Observation>
<pmlcore:Observation>
  <pmlcore:DateTime>2002-11-06T12:00:00-06:00</pmlcore:DateTime>
  <pmlcore:Data>
    <pmlcore:XML>
      <TemperatureReading xmlns="http://sensor.example.org/">
        <Unit>Celsius</Unit>
        <Value>5.8</Value>
      </TemperatureReading>
    </pmlcore:XML>
  </pmlcore:Data>
</pmlcore:Observation>
</pmlcore:Sensor>
</pmlcore:Tag>
</pmlcore:Observation>
</pmlcore:Sensor>
```

#### 5.1.5. ID Element

The **ID** element is defined in the 'Identifier.xsd' PML schema. It is of the type 'Identifier'. 'Identifier.xsd' is a universal structure schema and is used by other schemas like the PML Core schema. PML Core reuses the **ID** element from 'Identifier.xsd' to define the PML Core sensor data model.

The EPC™ is the default identification scheme to uniquely identify sensors and tags. The use of other identification scheme is supported, but is not encouraged.

If alternate identification schemes are to be used, the scheme must be labeled using the XML attributes of the identifier type.

If no other scheme is indicated, the EPC™ identification scheme must be used. The EPC™ must be represented as a URI as specified in [EPC™] or later versions of this specification.

The XML format of the identifier, which is of the [XSD] 'string' data type, facilitates this representation. The **ID** element has the following attributes

- an optional **schemeID** attribute
- an optional **schemeAgencyID** attribute
- an optional **schemeVersionID** attribute
- an optional **schemeURI** attribute

The actual format of the identifier code is the [XSD] 'string' data type without any restriction on its length.

#### 5.1.5.1. schemeID ATTRIBUTE

The schemeID attribute specifies the identifier of the identification scheme.

#### 5.1.5.2. schemeAgencyID ATTRIBUTE

The schemeAgencyID attribute specifies the identifier of the agency that maintains the identification scheme.

#### 5.1.5.3. schemeVersionID ATTRIBUTE

The schemeVersionID attribute specifies the version number of the identification scheme.

#### 5.1.5.4. schemeURI ATTRIBUTE

The schemeURI attribute specifies the Uniform Resource Identifier that identifies where the Identification Scheme is located.

#### 5.1.5.5. SAMPLE XML FOR ID ELEMENT

Sample showing the use of the EPC™ identification scheme:

```
<pmlcore: Sensor>
  <pmluid:ID>urn:epc:1:4.16.36</pmluid:ID>
  <pmlcore:Observation>
    <pmlcore:DateTime>2002-11-06T13:04:34-06:00</pmlcore:DateTime>
    <pmlcore:Tag>
      <pmluid:ID>urn:epc:1:2.24.400</pmluid:ID>
    </pmlcore:Tag>
    <pmlcore:Tag>
      <pmluid:ID>urn:epc:1:2.24.401</pmluid:ID>
    </pmlcore:Tag>
  </pmlcore:Observation>
</pmlcore:Sensor>
```

Sample showing the use of other identification schemes than the EPC™:

```
<pmlcore:Sensor>
  <pmluid:ID schemeID="MyScheme" schemeAgencyID="SomeCompany"
  schemeVersionID="v1">10023453</pmluid:ID>
  <pmlcore:Observation>
    <pmlcore:DateTime>2002-11-06T13:04:34-06:00</pmlcore:DateTime>
    <pmlcore:Tag>
      <pmluid:ID schemeID="MyScheme" schemeAgencyID="SomeCompany"
      schemeVersionID="v1">21114444</pmluid:ID>
    </pmlcore:Tag>
    <pmlcore:Tag>
      <pmluid:ID schemeID="MyScheme" schemeAgencyID="SomeCompany"
      schemeVersionID="v1">21114400</pmluid:ID>
    </pmlcore:Tag>
  </pmlcore:Observation>
</pmlcore:Sensor>
```

## 6. APPENDIX

### 6.1. XML Schemas

#### 6.1.1. PmlCore.xsd

```

<?xml version="1.0" encoding="UTF-8"?>
<schema
targetNamespace="urn:autoid:specification:interchange:PMLCore:xml:schema:1"
xmlns="http://www.w3.org/2001/XMLSchema"
xmlns:autoid="http://www.autoidcenter.org/2003/xml"
xmlns:pmlcore="urn:autoid:specification:interchange:PMLCore:xml:schema:1"
xmlns:pmluid="urn:autoid:specification:universal:Identifier:xml:schema:1"
elementFormDefault="qualified" attributeFormDefault="unqualified"
version="1.0">
  <import namespace="urn:autoid:specification:universal: Identifier:
xml:schema:1"
schemaLocation="../Universal/Identifier.xsd"/>
  <annotation>
    <documentation>
      <autoid:copyright>Copyright ©2003 Auto-ID Center, All Rights
Reserved.</autoid:copyright>
      <autoid:disclaimer>Auto-ID Center, its members, officers, directors,
employees, or agents shall not be liable for any injury, loss, damages,
financial or otherwise, arising from, related to, or caused by the
use of this document. The use of said document shall constitute your
express consent to the foregoing exculpation.</autoid:disclaimer>
      <autoid:program>Auto-ID version 1.0</autoid:program>
      <autoid:purpose>PML Core Specification version 1.0</autoid:purpose>
    </documentation>
  </annotation>
  <element name="Sensor" type="pmlcore:SensorType"/>
  <complexType name="AnyXMLContentType">
    <annotation>
      <documentation>
        <autoid:definition>The AnyXMLContentType provides localized openness
        </autoid:definition>
      </documentation>
    </annotation>
    <sequence>
      <any namespace="##any" processContents="skip">
        <annotation>
          <documentation>
            <autoid:definition>Any content</autoid:definition>
          </documentation>
        </annotation>
      </any>
    </sequence>
  </complexType>
  <complexType name="DataType">
    <annotation>

```

```

<documentation>
  <autoid:definition>The Data element holds text, binary or XML
  data.</autoid:definition>
</documentation>
</annotation>
<choice>
  <element name="Text" type="string">
    <annotation>
      <documentation>
        <autoid:definition>Text value</autoid:definition>
      </documentation>
    </annotation>
  </element>
  <element name="Binary" type="hexBinary">
    <annotation>
      <documentation>
        <autoid:definition>Binary value</autoid:definition>
      </documentation>
    </annotation>
  </element>
  <element name="XML" type="pmlcore:AnyXMLContentType">
    <annotation>
      <documentation>
        <autoid:definition>The XML element holds any XML elements
        the instance author would like to include. It is provided
        to enable localized openness and to allow instance document
        authors to create instance documents containing elements
        above and beyond what is specified by the PML CORE schema<
        /autoid:definition>
      </documentation>
    </annotation>
  </element>
</choice>
</complexType>
<complexType name="ObservationType">
  <annotation>
    <documentation>
      <autoid:definition>Information related to an observation/measurement
      by a sensor in the EPC Network. Observations represent measurements
      by the sensor. They associate the actual observed data with the
      sensor.</autoid:definition>
    </documentation>
  </annotation>
  <sequence>
    <element ref="pmluid:ID" minOccurs="0">
      <annotation>
        <documentation>
          <autoid:definition>The observation ID element is a number
          assigned to this specific observation.</autoid:definition>
        </documentation>
      </annotation>
    </element>
  </sequence>
</complexType>

```

```
</element>
<element name="DateTime" type="dateTime">
  <annotation>
    <documentation>
      <autoid:definition>The Observation DateTime element denotes
        the date and time stamp when the observation was made.</autoid:
        definition>
    </documentation>
  </annotation>
</element>
<element name="Command" type="string" minOccurs="0">
  <annotation>
    <documentation>
      <autoid:definition>The observation command element denotes
        the command was issued to the sensor to trigger the
        observation.</autoid:definition>
    </documentation>
  </annotation>
</element>
<element name="Tag" type="pmlcore:TagType" minOccurs="0"
maxOccurs="unbounded">
  <annotation>
    <documentation>
      <autoid:definition>The Observation Tag element denotes tags
        observed by a sensor as part of the observation.</autoid:
        definition>
    </documentation>
  </annotation>
</element>
<element name="Data" type="pmlcore:DataType" minOccurs="0"
maxOccurs="unbounded">
  <annotation>
    <documentation>
      <autoid:definition>The Observation Data element denotes any
        data captured by the sensors as part of the observation.</autoid:
        definition>
    </documentation>
  </annotation>
</element>
</sequence>
</complexType>
<complexType name="SensorType">
  <annotation>
    <documentation>
      <autoid:definition>Information related to a sensor in the EPC
        Network. A sensor is any device that is capable of making
        measurements e.g. RFID readers, temperature sensors, humidity
        sensors.</autoid:definition>
    </documentation>
  </annotation>
</complexType>
```



```
        <autoid:definition>The Tag Sensor element denotes any sensor
        that is mounted on the tag</autoid:definition>
    </documentation>
</annotation>
</element>
</sequence>
</complexType>
</schema>
```

### 6.1.2. Identifier.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<schema targetNamespace="urn:autoid:specification:universal:Identifier:xml:
schema:1"
xmlns:pmluid="urn:autoid:specification:universal:Identifier:xml:schema:1"
xmlns:autoid="http://www.autoidcenter.org/2003/xml"
xmlns="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
attributeFormDefault="unqualified" version="1.0">
    <annotation>
        <documentation>
            <documentation>
                <autoid:copyright>Copyright ©2003 Auto-ID Center, All Rights
                Reserved.</autoid:copyright>
                <autoid:disclaimer>Auto-ID Center, its members, officers,
                directors, employees, or agents shall not be liable for any
                injury, loss, damages, financial or otherwise, arising from,
                related to, or caused by the use of this document. The use of
                said document shall constitute your express consent to the
                foregoing exculpation.</autoid:disclaimer>
                <autoid:program>Auto-ID version 1.0</autoid:program>
                <autoid:purpose>PML Core Specification version 1.0</autoid:purpose>
            </documentation>
        </documentation>
    </annotation>
    <element name="ID" type="pmluid:IdentifierType"/>
    <annotation>
        <documentation>
            <autoid:definition>A reusable element of type
            'IdentifierType' </autoid:definition>
        </documentation>
    </annotation>
    <complexType name="IdentifierType">
        <annotation>
            <documentation>
                <autoid:definition>A character string to identify and distinguish
                uniquely, one instance of an object in an identification scheme
                from all other objects within the same scheme</autoid:definition>
            </documentation>
        </annotation>
        <simpleContent>
```

```

<extension base="token">
  <attribute name="schemeID" type="token" use="optional">
    <annotation>
      <documentation>
        <autoid:definition>The identifier of the identification
          scheme</autoid:definition>
      </documentation>
    </annotation>
  </attribute>
  <attribute name="schemeAgencyID" type="token" use="optional">
    <annotation>
      <documentation>
        <autoid:definition>The identifier of the agency that
          maintains the identification scheme</autoid:definition>
      </documentation>
    </annotation>
  </attribute>
  <attribute name="schemeVersionID" type="token" use="optional">
    <annotation>
      <documentation>
        <autoid:definition>The version of the identification
          scheme</autoid:definition>
      </documentation>
    </annotation>
  </attribute>
  <attribute name="schemeURI" type="anyURI" use="optional">
    <annotation>
      <documentation>
        <autoid:definition>The Uniform Resource Identifier that
          identifies where the Identification Scheme is located</
          autoid:definition>
      </documentation>
    </annotation>
  </attribute>
</extension>
</simpleContent>
</complexType>
</schema>

```

## 6.2. XML Instance Files

The XML instance files below demonstrate usage scenarios for basic data acquisition in the Auto-ID infrastructure, which is the data captured by sensors. They illustrate observations made by different types of sensor devices and the different types of data that may be captured using PML Core messaging between 2 PML enabled systems in the EPC™ Network System. Data communication between Savant™ and EPC™ Information Service or Enterprise Applications in the EPC™ Network system would be a typical PML Core messaging scenario. These XML Instance files are based on the PML Core schema documented in the previous section and are provided to illustrate the use of the same.

### 6.2.1. RFID Reader and Tags

At the very heart of the EPC™ Network System, RFID readers detect tags, which are identified by their EPC™. This example demonstrates how multiple tag read by a RFID Reader are represented.

#### 6.2.1.1. RFIDReaderAndTags.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<pmlcore:Sensor
xmlns:pmlcore="urn:autoid:specification:interchange:PMLCore:xml:schema:1"
xmlns:pmluid="urn:autoid:specification:universal:Identifier:xml:schema:1"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:autoid:specification:interchange:PMLCore:xml:schema:1
../SchemaFiles/Interchange/PMLCore.xsd">
  <pmluid:ID>urn:epc:1:4.16.36</pmluid:ID>
  <pmlcore:Observation>
    <pmluid:ID>00000001</pmluid:ID>
    <pmlcore:DateTime>2002-11-06T13:04:34-06:00</pmlcore:DateTime>
    <pmlcore:Command>READ_PALLET_TAGS_ONLY</pmlcore:Command>
    <pmlcore:Tag>
      <pmluid:ID>urn:epc:1:2.24.400</pmluid:ID>
    </pmlcore:Tag>
    <pmlcore:Tag>
      <pmluid:ID>urn:epc:1:2.24.401</pmluid:ID>
    </pmlcore:Tag>
    <pmlcore:Tag>
      <pmluid:ID>urn:epc:1:2.24.402</pmluid:ID>
    </pmlcore:Tag>
    <pmlcore:Tag>
      <pmluid:ID>urn:epc:1:2.24.403</pmluid:ID>
    </pmlcore:Tag>
    <pmlcore:Tag>
      <pmluid:ID>urn:epc:1:2.24.404</pmluid:ID>
    </pmlcore:Tag>
  </pmlcore:Observation>
</pmlcore:Sensor>
```

The following example illustrates the use of PML Core for the same scenario as mentioned above. The only difference is that the default identification scheme, the EPC, is not used.

#### 6.2.1.2. RFIDReaderAndTags2NoEPC.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<pmlcore:Sensor
xmlns:pmlcore="urn:autoid:specification:interchange:PMLCore:xml:schema:1"
xmlns:pmluid="urn:autoid:specification:universal:Identifier:xml:schema:1"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:autoid:specification:interchange:PMLCore:xml:schema:1
../SchemaFiles/Interchange/PMLCore.xsd">
  <pmluid:ID schemeID="MyScheme" schemeAgencyID="http://sensor.example.org/"
schemeVersionID="v1">10023453</pmluid:ID>
  <pmlcore:Observation>
```

```

    <pmlcore:DateTime>2002-11-06T13:04:34-06:00</pmlcore:DateTime>
    <pmlcore:Tag>
      <pmluid:ID schemeID="MyScheme"
schemeAgencyID="http://sensor.example.org/"
schemeVersionID="v1">21114444</pmluid:ID>
    </pmlcore:Tag>
    <pmlcore:Tag>
      <pmluid:ID schemeID="MyScheme"
schemeAgencyID="http://sensor.example.org/"
schemeVersionID="v1">21114400</pmluid:ID>
    </pmlcore:Tag>
  </pmlcore:Observation>
</pmlcore:Sensor>

```

### 6.2.2. RFID Reader and Tags with Data

Future generations of Auto-ID Center tags might also feature additional data that applications can read from and write to. When an RFID reader detects such memory tags, the tag ID and the data can be made available. This example illustrates the tag ID and data read from such a tag using a RFID Reader.

#### 6.2.2.1. RFIDReaderAndTagsWithMemory.xml

```

<?xml version="1.0" encoding="UTF-8"?>
<pmlcore:Sensor
xmlns:pmlcore="urn:autoid:specification:interchange:PMLCore:xml:schema:1"
xmlns:pmluid="urn:autoid:specification:universal:Identifier:xml:schema:1"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:autoid:specification:interchange:PMLCore:xml:schema:1
../SchemaFiles/Interchange/PMLCore.xsd">
  <pmluid:ID>urn:epc:1:4.16.36</pmluid:ID>
  <pmlcore:Observation>
    <pmluid:ID>00000001</pmluid:ID>
    <pmlcore:DateTime>2002-11-06T13:04:34-06:00</pmlcore:DateTime>
    <pmlcore:Tag>
      <pmluid:ID>210000A8900016F000169DC1</pmluid:ID>
      <pmlcore:Data>
        <pmlcore:XML>
          <EEPROM xmlns="http://sensor.example.org/">
            <FamilyCode>12</FamilyCode>
            <ApplicationIdentifier>123</ApplicationIdentifier>
            <Block1>FFA0456F</Block1>
            <Block2>00000000</Block2>
          </EEPROM>
        </pmlcore:XML>
      </pmlcore:Data>
    </pmlcore:Tag>
  </pmlcore:Observation>
</pmlcore:Sensor>

```

### 6.2.3. RFID Reader and Tags with mounted Sensors

It is understood that with the evolution of the EPC™ Network system different types of tags will evolve. The example below shows how the data resulting from the following scenario can be represented using PML Core: A temperature sensor is mounted on an active tag, which measures the temperature at certain time intervals and stores the data. Once the tag is in the vicinity of the RFID reader the tag transmits its ID and the data observed by the sensor

#### 6.2.3.1. RFIDReaderAndTagsWithSensor.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<pmlcore:Sensor
xmlns:pmlcore="urn:autoid:specification:interchange:PmlCore:xml:schema:v1_0"
xmlns:pmlunv="urn:autoid:specification:universal:ComplexType:Identifier:xml:s
chema:v1_0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:autoid:specification:interchange:PmlCore:xml:schema:v
1_0 PmlCore_1.xsd">
  <pmlunv:ID>urn:epc:1:4.16.36</pmlunv:ID>
  <pmlcore:Observation>
    <pmlunv:ID>00000001</pmlunv:ID>
    <pmlcore:DateTime>2002-11-06T13:04:34-06:00</pmlcore:DateTime>
    <pmlcore:Tag>
      <pmlunv:ID>urn:epc:1:2.24.400</pmlunv:ID>
      <pmlcore:Sensor>
        <pmlunv:ID>urn:epc:1:12.8.128</pmlunv:ID>
        <pmlcore:Observation>
          <pmlcore:DateTime>2002-11-06T11:00:00-06:00</pmlcore:DateTime>
          <pmlcore:Data>
            <pmlcore:XML>
              <TemperatureReading xmlns="http://sensor.example.org/">
                <Unit>Celsius</Unit>
                <Value>5.3</Value>
              </TemperatureReading>
            </pmlcore:XML>
          </pmlcore:Data>
        </pmlcore:Observation>
        <pmlcore:Observation>
          <pmlcore:DateTime>2002-11-06T12:00:00-06:00</pmlcore:DateTime>
          <pmlcore:Data>
            <pmlcore:XML>
              <TemperatureReading xmlns="http://sensor.example.org/">
                <Unit>Celsius</Unit>
                <Value>5.3</Value>
              </TemperatureReading>
            </pmlcore:XML>
          </pmlcore:Data>
        </pmlcore:Observation>
      </pmlcore:Sensor>
    </pmlcore:Tag>
  </pmlcore:Observation>
</pmlcore:Sensor>
```

#### 6.2.4. Sensor and Data

The EPC™ Network will not only consist of tags and readers, but also of sensors that monitor the environment and augment the data from the automatic identification sensors. The following example shows how observations made by a particular sensor can be represented as a simple data blob in hexbinary notation.

##### 6.2.4.1. SensorAndData.xml

```
<pmlcore:Sensor
xmlns:pmlcore="urn:autoid:specification:interchange:PMLCore:xml:schema:1"
xmlns:pmluid="urn:autoid:specification:universal:Identifier:xml:schema:1"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:autoid:specification:interchange:PMLCore:xml:schema:1
../SchemaFiles/Interchange/PMLCore.xsd">
  <pmluid:ID>urn:epc:1:4.16.36</pmluid:ID>
  <pmlcore:Observation>
    <pmlcore:DateTime>2002-11-06T13:04:34-06:00</pmlcore:DateTime>
    <pmlcore:Data>
      <pmlcore:Binary>0FB8A0F5CB0F11000FB8A0F5CB0F11000FB8A0F5CB0F1100</
      pmlcore:Binary>
    </pmlcore:Data>
  </pmlcore:Observation>
</pmlcore:Sensor>
```

### 6.3. Identifier representation within PML Core

The following representations of an identifier code were considered before the one mentioned above was chosen (listed above as the first option).

- An identifier format that has no constraints on the type or the number of characters must be used. Attributes are used to specify the type of identification scheme as e.g. recommended in ebXML Core Components Methodology.

The advantage of this approach is that it allows for making the EPC™ the default identification scheme, but it also permits the use of other identification scheme if they are properly labeled. It hence fulfills the corresponding requirement, and that updates to the available identification schemes can be made via code list rather than updates to the actual schemas. It also facilitates the use of the URI representations of the Electronic Product Code™ as outlined in “URI representations of the Electronic Product Code™”.

This approach does, however, not use the built-in data type structures of the [XSD] schema language, which would permit an automatic validity check of an identifier instance.

- A common data type for all EPC™ e.g. using the hexbinary data type from [XSD] with no limit on the actual number of bits represented.

This approach represents all Electronic Product Codes™, while the actual type of EPC™ used is still only available to the application developer by inspecting the header bit of the EPC™ or looking up the appropriate attribute of the identifier field. It restricts the use of PML Core to EPC™ based identifiers.

- A different data type for each representation of the Electronic Product Code™ e.g. a 96-bit version. This approach makes full use of the type checking possibilities of [XSD] parsers. However, new EPC™ representations require updates to the schemas and there is no flexibility with respect to other identification schemes.
- A structured, complex data type which maps the EPC™ structure into separate fields.

While this approach relieves the application developer from parsing the appropriate EPC™ for certain data bits e.g. the object class, it results in an overhead on the data routing side because all EPCs™ need to be converted into the various fields. Flexibility is difficult to achieve because later updates to the EPC™ format require updates to all places where EPCs™ gathered from RFID readers are converted into XML messages. Although this format might be useful for certain other usage scenarios, it is not recommended for the sensor observations scenarios addressed by PML Core, since the unique identification rather than the coding feature provided by the EPC™ is most needed here.

## 7. REFERENCES

1. **C. Floerkemeier & R. Koh, “Physical Mark-Up Language Update”.**  
Technical Memo, MIT-AUTOID-TM-006, July 2002.  
  
[UST]
2. **RosettaNet® Universal Structures.**  
  
[NSSM]
3. **RosettaNet® Namespace Specification and Management.**  
  
[XMLDG]
4. **RosettaNet® XML Design Guidelines.**  
  
[XSD]
5. **XML Schema Part 2, “Datatypes W3C Recommendation”.**  
02 May 2001 (<http://www.w3.org/TR/xmlschema-2/>).  
  
[RFC 2119]
6. **Key Words for use in RFCs to Indicate Requirement Levels.**  
Internet Engineering Task Force, March 1997 (<http://www.rfc-editor.org/rfc/rfc2119.txt>).  
  
[RFC 2141]
7. **URN Syntax.**  
May 1997, <http://www.ietf.org/rfc/rfc2141.txt>.  
  
[EPC]
8. **M. Mealling & Ken Traub, “The URI Representation of the Electronic Product Code and Related Types”.**  
Working Draft Version, 12 August 2003.

