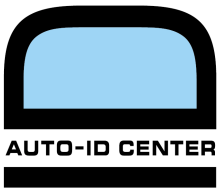
Published June 1, 2001. Distribution Restricted to Sponsors until September 1, 2001



WHITE PAPER

The Physical Markup Language

Core Components: Time and Place

David L. Brock, Timothy P. Milne,
Yun Y. Kang & Brendon Lewis

MIT AUTO-ID CENTER MASSACHUSETTS INSTITUTE OF TECHNOLOGY, 77 MASSACHUSETTS AVENUE, BUILDING 3-449G, CAMBRIDGE, MA 02139-4307

ABSTRACT

The Physical Markup Language (PML) is designed to be a common "language" for describing physical objects, processes and environments. Clearly it is difficult to describe the physical world with sufficient detail to meet the need of every company, industry or organization. However, there are common characteristics of physical objects that nearly everyone can agree on. Objects have physical properties – volume and mass. They often have structure – assemblies and elements. They are owned and traded among companies and individuals. They exist in time and space. The core components of the Physical Markup Language are intended to capture these most basic physical properties of objects and environments. In this paper, we will describe the elements that define simple configuration, location, time and measurement.[†] Beyond these elements, we will extend the depth and complexity of object description with future development of the Physical Markup Language.

WHITE PAPER

The Physical Markup Language

Core Components: Time and Place

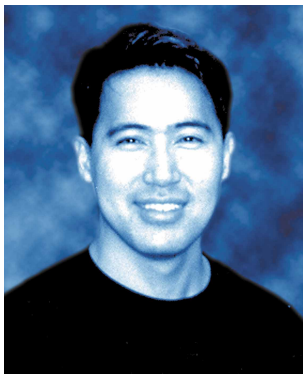
Biography

by David L. Brock
Co-Director

Dr. David Brock received Bachelors degrees in theoretical mathematics and mechanical engineering from MIT, and his Masters and Ph.D. degrees from the Department of Mechanical Engineering at MIT with an affiliation to the Artificial Intelligence Lab. He is currently a Principal Research Scientist in the Laboratory for Manufacturing and Productivity and Co-Director of the MIT Auto-ID Center. Dr. Brock is also the Founder of Brock Rogers Surgical, a manufacturer of robotic medical devices. Dr. Brock has worked with a number of organizations including the Artificial Intelligence Laboratory, the Massachusetts Eye and Ear Infirmary, DARPA, Lockheed-Martin, Loral, BBN and Draper Laboratories.

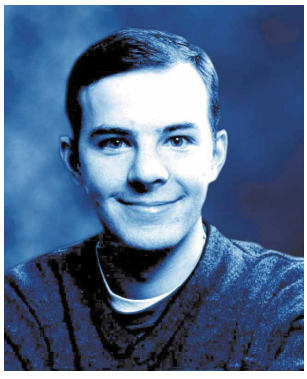
by Timothy P. Milne
Ph.D. Candidate

Timothy Milne received his Bachelors and Masters degrees from Brigham Young University where he graduated with honors from the department of Mechanical Engineering. At BYU he was involved in the Computer Aided Design (CAD) and Computer Aided Geometric Design (CADG) labs. His Masters thesis presented a new method for topologically mapping arbitrary N-Genus surfaces to single planar domains. Following graduation in 1994, Milne developed CAD software for various industries including the Aerospace Corp., Rhythm and Hues Studios and Varimetrix Corp. Most recently he worked for Corrpro, a corrosion protection consulting firm, writing a software package for corrosion protection management.



by Yun Y. Kang
Ph.D. Candidate

Yun Kang's doctoral research at the Auto-ID Center involves design and control of supply chain management, with particular focus on the impact of automatic identification technology on today's supply chain. He has also closely worked with the sponsors in identifying the applications of automatic identification and developing business cases. Prior to joining the Auto-ID Center, Yun studied design theory for product and process development and its application in semiconductor photolithography machines. Yun received his Bachelors from The Cooper Union and Masters from MIT, both in mechanical engineering. He also has research experience in mathematical modeling, applied superconductivity, and automatic control systems.



by Brendon Lewis
Graduate Student

Brendon Lewis received his Bachelors of Science in Mechanical Engineering from Tufts University in 2000. He is a candidate to receive his Masters of Science degree in June 2002 at the Massachusetts Institute of Technology. Brendon joined the Auto-ID Center in the fall of 2000. His research interests include modeling, simulation, controls, and robotics. Brendon's current research with the center examines improvements in the supply chain through the use of Auto-ID Center technology. His work focuses on modeling and simulation of supply chain processes, and he is also part of a team that is developing the physical markup language.

WHITE PAPER

The Physical Markup Language

Core Components: Time and Place

Contents

The Physical Markup Language (PML) is designed to be a common "language" for describing physical objects, processes and environments. Clearly it is difficult to describe the physical world with sufficient detail to meet the need of every company, industry or organization. However, there are common characteristics of physical objects that nearly everyone can agree on. Objects have physical properties – volume and mass. They often have structure – assemblies and elements. They are owned and traded among companies and individuals. They exist in time and space. The core components of the Physical Markup Language are intended to capture these most basic physical properties of objects and environments. In this paper, we will describe the elements that define simple configuration, location, time and measurement.[†] Beyond these elements, we will extend the depth and complexity of object description with future development of the Physical Markup Language.

[†] This paper presents on-going development of the Physical Markup Language and is not intended as an initial specification. Date structures, syntax and semantics are likely to change in the course of development. However, the basic approach and the level of functionality will likely remain intact.

1. INTRODUCTION

The Physical Markup Language (PML) is intended to be a general, standard means for describing the physical world. Given the difficulty of such a task, we must carefully consider the objective of PML and its intended application.

The objective of PML is a simple, general language for describing physical objects for use in remote monitoring and control of the physical environment. Applications include inventory tracking, automatic transaction, supply chain management, machine control and object-to-object communication.

Our approach will be to develop a series of core components that span the breadth of application, and which are independent of vertical industries. We and space, and their movement and location are critical to commerce and industry. Products have attributed values of worth, cost, price and ownership.

With work presented here and subsequent development, we wish to provide a series of basic components that capture these basic properties. The present effort addresses simple object configuration, location, time and measurement.

The following sections include a background section outlining the development of PML in the context of our proposed 'intelligent infrastructure' [1]. The next section describes our approach to the language development. The Design section details some of the core components and provides examples of their application. The conclusion summarizes the results and presents near-term efforts to complete the core specifications.

2. BACKGROUND

2.1. The Intelligent Infrastructure

The Physical Markup Language exists as part of the 'intelligent infrastructure.' The intelligent infrastructure, which we envision, automatically and seamlessly links physical objects to each other, people and information through the global Internet [1,2]. This intelligent infrastructure has four major components: electronic tags, Electronic Product Code (EPC), Physical Markup Language (PML) and Object Naming Service (ONS) [3,4].

Electronic tags refer to a family of technologies that transfer data wirelessly between tagged objects and electronic readers. Radio Frequency Identification (RFID) tags, often used in "smart cards," have

small radio antennas, which transmit data over a short range [5]. The Motorola BiStatix™ tags, an Electromagnetic Identification (EMID) technology, uses capacitive coupling to transmit information [6]. Electronic tags, when coupled to a reader network, allow continuous racking and identification of physical resources. In order to access and identify tagged objects, a unique naming system was developed.

The Electronic Product Code (EPC) was conceived as a means to identify physical objects [3]. The EPC code was created to enumerate all objects and to accommodate current and future naming methods. The EPC code was intended to be universally and globally accepted as a means to link physical objects to the computer network, and to serve as an efficient information reference.

The Object Naming Service (ONS) is the **"**glue,**"** which links the Electronic Product Code (EPC) with its associated data file [7]. More specifically, the ONS is an automated networking service, which, when given an EPC number, returns a host addresses on which the corresponding data file is located. The ONS, currently under development, is based on the standard Domain Naming Service (DNS). When complete, the ONS will be efficient and scaleable, designed to handle the trillions of transactions that are expected.

Finally, the Physical Markup Language (PML) is intended to be the standard in which networked information about physical objects is written [4]. In one sense, all the complexity of describing and classifying objects has moved away from the object label and into the PML file. The formation of this language – together with the associated software tools and applications – is one of the most difficult aspects of this **"Internet of Things."**

2.2. PML Syntax

Rather than develop a separate syntax, PML uses the same format and structure as the eXtensible Markup Language (XML) [8,9]. The XML syntax is basically a set of tag elements used to demark data and provide some indication of meaning. For example,

```
<company_name>ACME, Co. </company_name>
```

indicates the string 'ACME, Co.' is a 'company name.' This allows the computer to more easily parse and present networked information. The XML specification is, of course, more involved than this. The language provides mechanisms for network links, data types and schema.

A schema – often termed a **meta-language** – is language for describing the form and structure of a language, which in the case of XML is the format of the tag elements [10,11,12]. PML therefore is essentially a specific schema of XML intended to describe physical objects and environments.

2.3. eCommerce Standards

There are a number of on-going and related efforts to develop languages for electronic commerce. These include the Electronic Data Interchange (EDI) specification that provides a collection of standard message formats and element dictionaries allowing businesses to exchange data through electronic messaging services. The Electronic Data Interchange is defined in the American National Standard X12 syntax [13] and the United Nations/Electronic Data Interchange for Administration, Commerce and Transport (UN/EDIFACT) syntax [14].

A newer ebXML initiative is a set of specifications that describe business processes and transactions, as well as the choreography of business documents between trading partners [15]. ebXML uses XML-based messages exchanged and translated through shared registries. ebXML is sponsored by the United Nations Centre for Trade Facilitation and Electronic Business (UN/CEFACT) and the Organization for the Advancement of Structure Information Standards (OASIS) [15].

The Universal Description, Discovery and Integration (UDDI) initiative intends to provide business with shared business, service and contact information through global registries [16]. Supported by IBM, Microsoft and Ariba, UDDI uses existing standards such as XML and the Simple Object Access Protocol (SOAP) [17] together with registries providing shared business contact information 'White Pages', business categories and organization 'Yellow Pages', and business process and service descriptions 'Green Pages.'

While these and other eCommerce initiatives may augment and enhance the Physical Markup Language and our proposed 'intelligent infrastructure,' they have a very different focus. The intention of PML is capture and describe the real-time state of the physical environment, while the eCommerce standards target the mechanisms and documentation of business transaction. Changing states of the physical environment, such as the arrival of a shipment, may of course trigger business events, such as an invoice. Therefore, we intend to PML to workd in cooperation and inconjuction with electronic business process standards.

3. APPROACH

3.1. Syntax

Our approach for the Physical Markup Language is to first use existing standards for syntax and data transmission, such as the eXtensible Markup Language (XML), the HyperText Transport Protocol (HTTP) and Transmission Control Protocol and Internet Protocol (TCP/IP). This provides a basic set of functionality and existing tools to build and develop PML applications.

3.2. Semantics

Rather than develop a meta-standard, in which disparate schema translate through a shared registries, PML will provide a single specification. Using a common, implied schema – such as the HyperText Markup Language (HTML) – translation between schemas is unnecessary, and reliable transmission and interpretation is provided. Furthermore, a single specification encourages 3rd party development of software viewers, editors and applications.

The Physical Markup Language will strive toward a **single** representation for any data element. In other words, if there are multiple ways to encode a data type, PML will arbitrarily select one. For example, of the myriad ways to encode date, PML will select only one. The idea is that data translation occurs when encoding or viewing, not during data interchange.

3.3. Data storage and management

Although we often refer to a PML 'file,' the data format may not necessarily be used to actually store the data. Since PML is a method to demark information for distribution, the actual content may reside

in any format inside the server (e.g. an SQL database, spreadsheet or even a 'flat' file). In other words, a company does not have to store information in the PML format to use PML. Companies will maintain data in existing formats with current procedures.

For example, an applet can respond to requires from the Internet via the Object Name Service (ONS), extract the necessary data and reformat it in the PML specification for transmission. This is similar to Dynamic HTML (DHTML), which reformats an HTML page based on a user's input.

Furthermore a PML 'file' may not be a file at all, but a collection of many files and transmissions from many different sources. Because of the inherently distributed nature of the physical environment, the PML 'file' may actually collate small PML fragments from as many different locations.

Therefore a PML 'file' may only exist during transmission. The data itself may be ephemeral – existing for a short time and discarded after use.

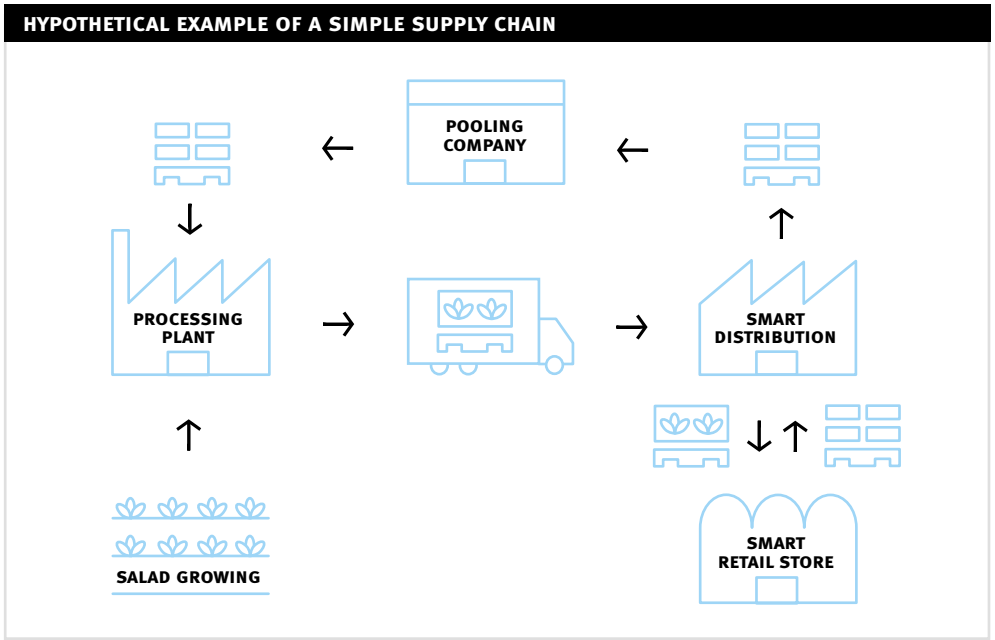
4. DESIGN

4.1. Overview

In the following sections, we will detail elements of the Physical Markup Language that describe configuration, location, time and measurement of physical objects.

In order to provide this information, PML provides a number of constructs and data types. These include a **data** element, which captures a 'snapshot' of the physical environment. A **node** element, described in the next section, conveys the hierarchical structure of physical systems. The **trace** element records not only where an object is, but also where the object was. The **entity** element contains information necessary to assign ownership and responsibility for physical objects. The final three sections **location**, **date** and **measure** specify the location of an object, time of observation and data measurement, such as weight

Figure 1. A hypothetical example of a simple supply chain. Salads are shipped to a retail store in reusable plastic containers on pallets both of which are managed by a container pooling company.



or temperature.
To illustrate these concepts, we provide a hypothetical example of a simple supply chain, which is shown in Figure 1. The grower and packager of bagged salad, **'Snappy Salads, Inc.'** ships produce to **'SMart Stores'** in reusable plastic containers (RPC) on pallets both of which are supplied by **'International Pooling Company (IPC)'**. All shipping and transport are provided by **'U.S. Transit, Co.'**.

4.2. Data

As items flow through the supply chain, it is often necessary to take a temporary inventory – an instantaneous record of the products in storage. We use a simple **data** element to contain a 'snapshot' of the physical state of the environment.

The **data** element will contain a timestamp; that is a time marker at the instant the data was gathered. We will use a **date** element, which is described later, to serve this purpose. The **data** element is essentially a container to house the data structures that are presented in the following sections. Electronic Product Codes (**EPC** elements), physical structures (**node** elements) and measurements (**measure** elements) are all contained within the **data** 'wrapper.'

Using the XML notation, the **data** element takes the following simple form:

```
<data>
    <date> … </date>
    …
</data>
```

The ellipses '…' indicate other PML elements or data forms that must be supplied. In the following sections, we will use this format to show the general form of the PML elements. We will also include examples to illustrate its application to supply chain management.

4.3. Hierarchy

When you purchase an item from the store, you may be actually buying a group of products that were made at different times, locations and even by different manufacturers. For example, you could purchase a razor handle by itself, or the handle with four blades. You could buy replacement blades in various configurations – all with or without a handle – or you could buy a promotional package that includes the handle, blades, shaving cream and aftershave. Regardless of the configuration, the handle and the blades are the same. The only difference is their relationship to other objects. This relationship of items is described by an object hierarchy. Since many physical systems have this hierarchical structure, we propose PML capture this information.

To describe a hierarchy, a common analogy is a tree whose elements are arranged as branches and leaves. A directory file structure, ancestors and descendents in a family line or – more generally – a connected, a-cyclic graph are other ways to describe the same concept.

The PML semantic will define an element called a **node**. There are a number of attributes associated with the node, the most important of which are the **label** and **EPC** attributes:

```
<node label="name" EPC="xx.xxxxxxx.xxxxxx.xxxxxxxxx">
</node>
```

The next requirement of a node is that it can contain zero or more nested nodes:

```
<node label="name" EPC="x … x">
    <node label="name" EPC="x … x">
        <node label="name" EPC="x … x">
        </node>
    </node>
    <node label="name" EPC="x … x"></node>
</node>
```

The node hierarchical syntax is simply a nested collection of node elements. These elements will contain other information or links to other PML files.

A node can function as either a parent or child or both. The ability of a node to contain another node is an example of recursion, which is an important feature when representing structure in the physical world.

There will be one, and only one, Root Node entity in a fully assembled PML file. The Root Node can contain any number of child nodes, but cannot be a child of another node, nor can it have any siblings. Not all the data will be in the same PML file, since a finished object may consist of parts from several manufacturers. Therefore, the PML file must be able to include objects from other PML files in a modular fashion.

In most relational hierarchies, the parent knows its first child and next sibling nodes, and a child knows its next sibling. This allows for easy traversal of the hierarchal family tree. This information will not be included in the PML specification, but will be left to the application to retrieve as necessary, since it is easily parsed and reconstructed. This is an important consideration since it eliminates the need to maintain and update related data as the relationships change over time.

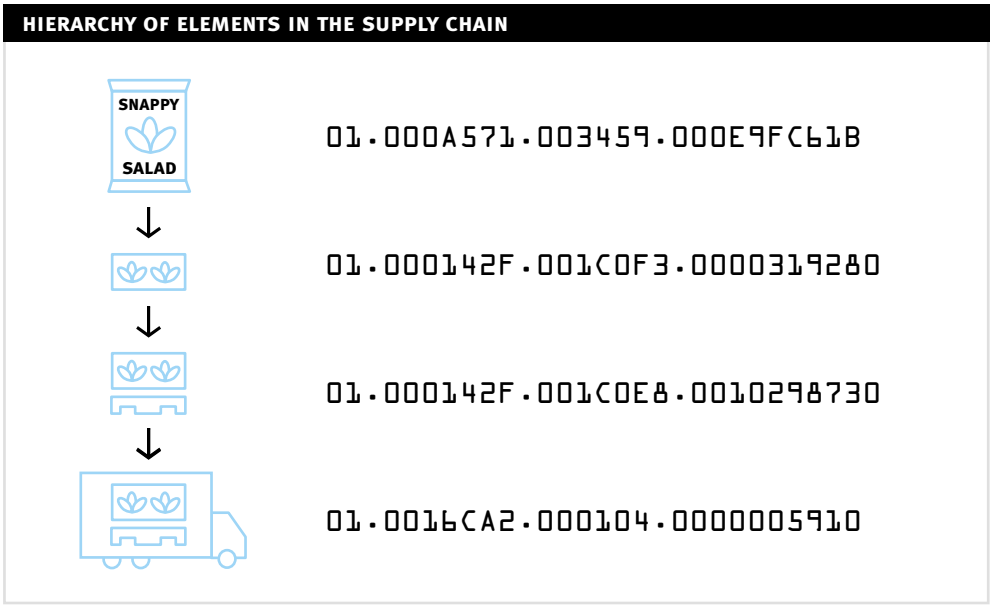
Example
The example shown in Figure 2 illustrates a simple hierarchy that may exist in the supply chain. A transport contains a number of pallets, each of which holds a stack of reusable plastic containers (RPCs). RPCs are used commonly to transport perishable foods, such as the lettuce bag shown in the figure.

The PML code segment that represents this organization is given below in Listing 1. With this PML listing we could find the identity of the truck from the bag of salad, or given the truck, the entire contents of its load.

Listing 1.
The PML node structure captures the organization of physical objects.

```
<node label="Truck" EPC="01.0016CA2.000104.0000005910">
                …
    <node label="Pallet" EPC="01.000142F.001C0E8.0010298730">
                …
        <node label="RPC" EPC="01.000142F.001C0F3.0000319280">
                …
            <node label="Salad" EPC="01.000A571.003459.000E9FC61B">
                …
            </node>
        </node>
    </node>
</node>
```

Figure 2. Elements in the supply chain may be represented as hierarchies. Here salad is packed in reusable plastic containers (RPCs), which are stacked on pallets placed in transport vehicles.



4.4. Trace

In many cases it is important to know not only where an object is, but also where an object was. The PML specification must provide a mechanism to trace the path of a product, case or shipment as it moves through the supply chain. Here we provide a **trace** specification, which records the movement history of an object.

The **trace** element includes one or more **steps** that indicate waypoints along the path followed by an object. Each **step** has a number of optional elements including an **owner**, **date** and **location**:

```
<trace>
    <step>
        <owner> … </owner>
        <location> … </location
        <date> … </date>
    </step>
    <step>
    …
    </step>
</trace>
```

The specification allows zero or more **owners** at each s**tep** in the **trace**. The **owners** represent the companies, organizations or individuals responsible for an item at each **step** in its movement.

The **owner** includes a **type** attribute indicating the 'type' of 'owner.' At any step in the supply chain, many companies, organizations and individuals are involved with a product. These include the legal owner, shipper, carrier, insurer, distributor, storage, wholesaler, retail, etc. Therefore the **owner** element has a **role** element, which may include values, such as "Owner," "Shipper," "Carrier," "Insurer," "Storage," "Wholesale," "Retail" and "Other." This list, of course, will evolve during the development of the specification:

```
<owner>
    <role>
        [e.g. "Owner", "Shipper", "Carrier", "Insurer",
            "Storage", "Wholesale", "Retail", etc.]
    </role>
    <entity> … </entity>
</owner>
```

The **owner** element also includes an **entity** element, which includes specific information about the legal entity – name, description, contact information, etc. This element will be described in more detail in a following section.

It is not necessary for the location of an object to have any relation to the owner of that object. Therefore we include with each **step** a **location**; that is the physical location of the object. The **location** specification will also be described in a later section.

Finally, each **step** includes zero or more **dates** indicating the time at which an item arrives or departs from a location, or changes 'ownership' from one party to another. Times and dates in PML files will have a simple and uniform specification. This is outlined in the **date** section of the paper. Putting it all together, the **trace** specification has the following structure:

```
<trace>
    <step>
        <owner>
                <role>string</role>
                <entity> … </entity>
            </owner>
            <owner>
                <role>string</role>
                <entity> … </entity>
            </owner>
        <location> … </location
        <date> … </date>
    </step>
    <step>
    …
    </step>
</trace>
```

Example

For example, a salad bag moves from a large bin to a reusable plastic container, which is moved onto a pallet. The pallet is loaded onto a truck and shipped to a distribution center where it is unloaded. From distribution center, the pallet is reloaded onto another truck bound for a local retailer. A portion of this process is illustrated in Figure 3 and captured in the PML file as shown in Listing 2.

Listing 2.

The **trace** element tracks the movement of the salad bag from the processing plant through the distribution center to the retail store. Also the **entity**, **location** and **date** elements are described in later sections. For clarity, the **entity** and **location** elements are not shown in detail here.

```
<trace>
    …
    <step>
        <owner>
               <role>"Owner"</role>
               <entity>…"Snappy Salads, Inc."…</entity>
           </owner>
        <location>…"Processing Plant"…</location>
        <date type="Release">3938406045</date>
    </step>
    <step>
        <owner>
               <role>"Shipper"</role>
               <entity>…"U.S. Transit…"</entity>
           </owner>
        <location> …"Processing Plant"…</location>
        <date type="Acquire">3938406045</date>
    </step>
    <step>
        <owner>
               <role>"Shipper"</role>
               <entity>"U.S. Transit"</entity>
           </owner>
        <location> …"DC"…</location>
        <date type="Release">3956197489</date>
    </step>
    <step>
        <owner>
               <role>"Owner"</role>
               <entity>…"SMart Stores."…</entity>
           </owner>
        <location>…"DC"…</location>
        <date type="Acquire"> 3956317456</date>
    </step>
    <step>
        <owner>
               <role>"Owner"</role>
               <entity>…"SMart Stores."…</entity>
           </owner>
        <location>…"DC"…</location>
        <date type="Release"> 3957037654</date>
    </step>
    <step>
        <owner>
               <role>"Owner"</role>
               <entity>…"SMart Stores."…</entity>
           </owner>
        <location>…"Retail"…</location>
        <date type="Acquire"> 3959197455</date>
    </step>
</trace>
```

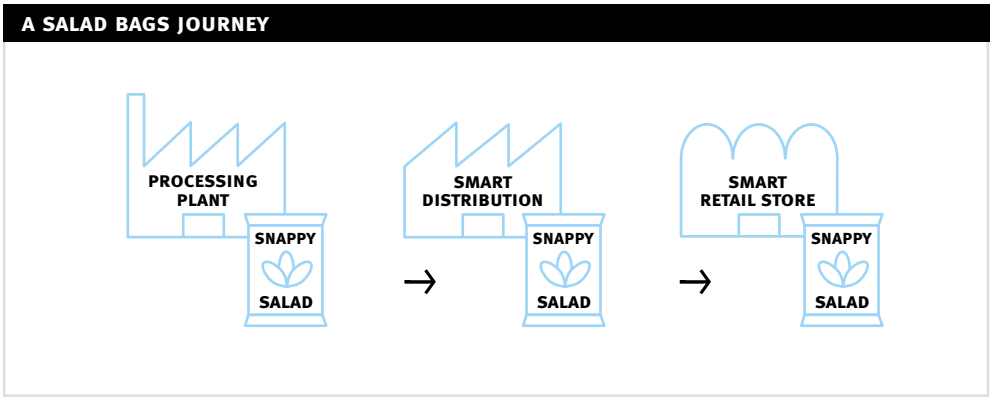


Figure 3. The salad bag travels from the processing plant through the distribution center to the retail store.

4.5. Entity

Almost every physical object is either owned or managed by someone or some group. Furthermore the transition in ownership between entities defines most commerce. Therefore we must include a mechanism within the Physical Markup Language to describe entities – whether people, companies or organizations – that own or manage physical objects.

We propose the concept of an **entity**, which may be a person, company, organization or group, which owns, manages or oversees physical assets. The **entity** will have a number of inherent characteristics as well as a list of **associations**.

Individuals, companies and organizations may have many **associations**. For example, a person may have more than one office, assistant, job, role, responsibility, affiliation or even home. The idea is to capture the network of links among individuals and companies in order to afford more accurate, timely and relevant communication.

We should note the purpose of the **entity** element is to unambiguously define ownership and responsibility for a physical resource and to establish a communication **channel**. Clearly, the information needed to more fully describe an individual, company or organization is not included here. There are, however,
a number of initiatives to build more complete descriptions for electronic commerce [14,15,16].

The **entity** element includes a **type** element. This will contain a string which defines the 'type' of **entity**, such as a "Person," "Company," "Corporation," "organization," etc.

The next element is the name of the **entity**; that is a human-readable, alphanumeric designation. In the case of an individual, we provide subordinate elements for **title**, **first**, **last**, **middle**, **suffix** and **alternate** (for nicknames or common names in the case of companies). For each of these elements, we define an optional **present** attribute – either "Full" or "Initial" – which describes how the name should be presented:

```
<entity>
    <type> String </type>
    <name>
        <title> String </type>
        <first present = [“Full,"Initial"]> String </first>
        <middle present = [“Full,"Initial"]> String </middle>
```

```
        <last present = [“Full,”Initial”]> String </last>
        <suffix> String </suffix>
        <alternate> String </alternate>
    </name>
    …
</entity>
```

Entities have zero or more **associations**. **Associations** represent links to communication **channels**, such as PDA, phone, FAX, email, etc., **addresses** of homes, offices, warehouses, factories, distribution centers, loading docks, etc., other **entities**, such as assistants, managers, departments, companies, affiliations or organizations.

Each **association** may include a **type** element, which represents the ‘type’ of association, such as “Personal,” “Home,” “Work,” “Affiliation,” etc. A **name** element contains a human readable string designating the association. A **role** element describes the ‘role’ an entity has within the association, such as “Foreman,” “Driver,” “Quality Control Manager,” etc. The **association** also includes zero or more **addresses**, communication **channels** and other **entities**:

```
<entity>
    …
    <association>
        <name> String </name>
        <type> String </type>
        <role> String </role>
        <address> … </address>
        …
        <channel> … </channel>
        …
        <entity> … </entity>
        …
    </association>
    …
</entity>
```

The **address** element includes a series of subordinate elements defining a physical location of the association. These include the following optional elements: **name**, **room**, **suite**, **number**, **building**, **street**, **city**, **county**, **province**, **state**, **region**, **country** and **code**. The **address** element also includes zero or more **channels**, since a physical location may have a communication **channel** separate from the entity. For example, a loading dock may have a telephone, which is not linked to any individual. The **address** specification is summarized as follows:

```
<adress>
    <name> String </name>
    <room> String </room>
    <suite> String </suite>
    <number> String </number>
    <building> String </building>
    <street> String </street>
    <city> String </city>
    <county> String </county>
    <province> String </province>
```

```
    <state> String </state>
    <region> String </region>
    <country> String </country>
    <code> String </code>
    <channel> … </channel>
</address>
```

Finally, a communication **channel** represents a method and mode of communication with the **entity** through a particular **association**. The **channel** includes a **type** attribute – "Voice," "Email," "FAX," "Mobile," "Pager," "Web," "Radio," "Telex," "TTY/TDD" or "Other" – and a string representing the actual number or address.

```
<channel type=["Voice", "Email", "FAX", "Mobile", "Pager", "Web",
              "Radio", "Telex," "TTY/TDD"]>
String
</channel>
```

Example

The shipping manager of S.Mart Stores, 'Robert Mooreland,' has a number of **associations** including a personal communication system (a cell phone), an office and the receiving docks. The office, located in a regional distribution center, also has an **association** with the larger S.Mart Stores Corporation. In this way, ownership and responsibility of physical assets may be captured and traced within this data structure.

The illustration shown in Figure 4 graphically represents these links, as well as indicates the communication channels. Listing 3 provides a simple PML encoding of this information. An actual listing of all the links, address and communication channels would likely be much larger. The idea here is to more accurate capture often competing and disparate associations the inventory manager may have, and to provide a clear representation of responsibility and communication.

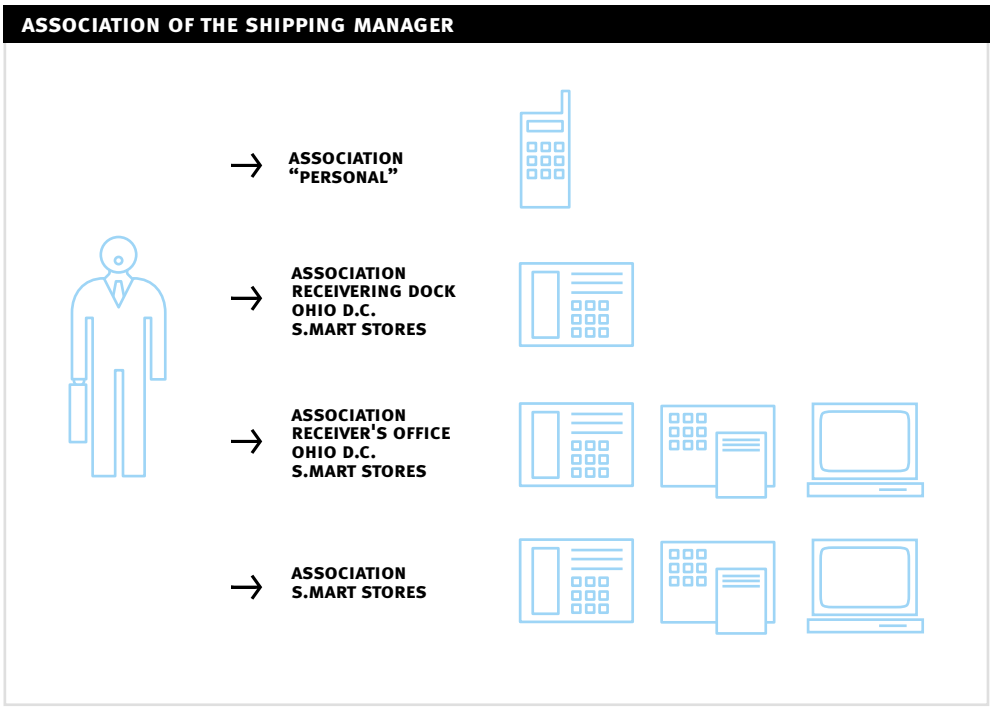


Figure 4. The receiving manager at the 'S.Mart' Distribution Center has a number of associations, including an office, loading dock and personal mobile phone.

Listing 3.
The 'Shipping Manager' has a number of associations, which include multiple addresses and communication channels.

```
<entity>
    <type>Person</type>
    <name>
        <title>Mr.</type>
        <first> Robert </first>
        <middle>Douglas </middle>
        <last>Mooreland</last>
    </name>

    <association>
        <type>Personal</type>
        <channel type="Mobile">513-632-8229</channel>
    </association>

    <association>
        <name> Receiving Dock</name>
        <type>Work</type>
        <role>Shipping Manager</role>
        <channel type="Voice">513-633-2938</channel>
        <address>
            <building>F19</building>
            <number>143</number>
            <street>Commerce Park Road</street>
            <city>Cincinnati</city>
            <state>Ohio</state>
            <country>USA</country>
            <code>45202-1579</code>
        </address>
        <entity> …"link S.Mart Stores"… </entity>
    </association>

    <association>
        <name> Shipping Manager Office</name>
        <type>Work</type>
        <role>Shipping Manager</role>
        <channel type="Voice">513-633-2942</channel>
        <channel type="Fax">513-633-2953</channel>
        <channel type="Email">rmooreld@smart.com</channel>

        <address>
            <building>D20</building>
            <number>143</number>
            <street>Commerce Park Road</street>
            <city>Cincinnati</city>
            <state>Ohio</state>
            <country>USA</country>
```

```
            <code>45202-1582</code>
        </address>
        <entity> …"link S.Mart Stores"… </entity>
    </association>
</entity>
```

4.6. Location

One of the most important questions to answer in the supply chain is “Where is my shipment?” PML must therefore define a specification for location. Although location is a seemingly simple concept it is somewhat difficult to represent accurately. Furthermore locations are usually referenced with respect so some other known object. For example, the ‘item is on the top shelf,’ or ‘pallet is at the front of the truck.’ Therefore we must include a reference frame for a location measurement.

We define a **location** element with an optional **EPC** attribute that defines the reference frame. In other words, the location of the object is measured from a coordinate frame in the **EPC** object. The **location** element includes one or more **measure** elements, which are defined in the following section. Finally, the **location** element may include another **location** element. In this way, a location may be recursively defined. For example, a pallet may be measured within a truck and the truck with respect to the route.

If the EPC attribute of the **location** element is omitted, the **measurements** are assumed to be with respect to a world coordinate frame. We define a canonical world reference in the same manner as the Global Positioning System (GPS) [18]. In this case, the three **measure** elements are latitude, longitude and altitude. The latitude is measured in meters from the equator, the longitude is meters from the prime meridian, and the altitude meters from the center of the earth:

```
<location>
    <measure label="latitude" m=1 accuracy=Integer>
        float
    </measure>
    <measure label="longitude" m=1 accuracy=Integer>
        float
    </measure>
    <measure label="altitude" m=1 accuracy=Integer>
        float
    </measure>
</location>
```

The general form for the **location** specification is

```
<location EPC="xx.xxxxxxx.xxxxxx.xxxxxxxxx">
        <measure> … </measure>
        …
        <measure> … </measure>
    <location EPC="xx.xxxxxxx.xxxxxx.xxxxxxxxx">
        …
    </location>
</location>
```

Example

Let's look at the movement of 'Snappy' packaged salads shown in Figure 3. The processing plant of the bagged salads is located in California and the distribution center is located in Ohio. When the 'U.S. Transit' truck loads the salads at the 'Snappy Salads Processing Center,' a step in the PML file indicates the acquire date, the transfer of ownership and the location of the bag of salad, as shown in Listing 4. The location is described using latitude, longitude and altitude. The 'U.S. Transit's' truck then arrives at 'S.Mart's' distribution center. The subsequent step indicates the release date, transfer of ownership and the location.

Listing 4.

Each step details the location of the packaged salad at a given moment in time. This movement in time is specified with a date, which is discussed later.

```
<trace>
    <step>
            <owner>
                <role>"Owner"</role>
                <entity>…"Snappy Salads, Inc."…</entity>
            </owner>
        <location>…"Processing Plant"…</location>
        <date type="Release">…</date>
    </step>
    <step>
            <owner>
                <role>"Shipper"</role>
                <entity>…"U.S. Transit…"</entity>
            </owner>

            <location>
                <measure label="latitude" m=1 accuracy=5>
                    4058808.538
                </measure>
                <measure label="longitude" m=1 accuracy=5>
                    625111.743
                </measure>
                <measure label="altitude" m=1 accuracy=5>
                    16.154
                </measure>
            </location>

            <date type="Acquire">…</date>
    </step>
    <step>
            <owner>
                <role>"Shipper"</role>
                <entity>"U.S. Transit"</entity>
            </owner>

            <location>
                <measure label="latitude" m=1 accuracy=5>
                    4058808.538
```

```
                </measure>
                <measure label="longitude" m=1 accuracy=5>
                    625111.743
                </measure>
                    16.154
                </measure>
            </location>

        <date type="Release">… </date>
    </step>

    <step>
        <owner>
            <role>"Owner"</role>
            <entity>…"S.Mart Stores."…</entity>
        </owner>
        <location>…"DC"…</location>
        <date type="Acquire"> … </date>
    </step>
</trace>
```

4.7. Date

Accurate time and date are critical for the supply chain. Shipping, delivery, receiving, transport and expiration dates must be carefully represented, recorded and transmitted to many interested parties including manufactures, distributors, retailers and consumers.

Clearly there are many ways to represent time and date – years, months, days, hours, minutes and seconds reference from 24 separate time zeros. There are also various methods for storing and displaying this information: June 1, 2001; 7-1-2001; 2001.7.1; Fri Jun 1, 2001 19:39:30, etc.

Consistent with our philosophy of simple representations for automated transmission, we propose a single **date** standard. A date is a duration measured from a given reference point. The selection of this reference point is somewhat arbitrary, but its existence is vital. Therefore we propose all dates are measured from January 1, 2000 midnight Greenwich Mean Time.

Units for time vary. Age is measured in years, plane flights in hours, television shows in minutes and stoplights in seconds. However, we propose a date measure in milliseconds. This should provide more than enough accuracy for most date measurements. Thus the date measurement in PML is the number of milliseconds that have passed since January 1, 2000 00:00:00.000 GMT.

Dates may assume both positive and negative values allowing measurement before January 1, 2000 – though presumably most PML files will produce dates after this. However, representing dates with a signed 64-bit integer will provide measurement until January 1, 292271023. That should be sufficient.

It is important to represent different types of dates. Dates of manufacture, acquisition, release, delivery, and expiration must be indicated with the date specification. Therefore we include a **type** attribute in the **date** element.

Clearly millisecond accuracy is unnecessary and misleading when specifying a delivery date. Therefore we include an optional **resolution** attribute. **Resolution** is an integer representing the number of digits to ignore in the **date** value. A **resolution** value of 3, for example, means the **date** value has an accuracy measured in seconds. A **resolution** value of 8 would correctly be interpreted as a day measurement.

The **date** element thus assumes the following form:

```
<date type=["String"] resolution=Integer>
Integer
</date>
```

Example
The example code shown in Listing 2 shows the package salad leaving the 'Snappy Salads' Processing Plant on March 1, 2001 at 6:01:145 and arrives at the distribution center in on March 3, 2001 at 7:26:489 – roughly two days later. The ownership transitions to 'S.Mart Stores' twenty minutes after arrival. The salads are only kept for two hours before their departure for the local 'S.Mart' retail store, where it arrives six hours later.

Although millisecond notion is difficult to perceive for a human reader, it is quite easy for the computer to understand.

4.8. Measurement

Physical states of matter are compared to known references. From cubits to nanometers from stones to dekagrams, multiples of common standards provide the means of communicating physical properties. We propose to a single standard of measure within the Physical Markup Language. This will simplify data communication and facilitate its presentation among disparate systems of measure using PML as a common medium.

Table 1. The seven PML base units assure mutual independent, unambiguous measurement [19].

BASE	NAME	SYMBOL
LENGTH	meter	m
MASS	kilogram	kg
TIME	second	s
CHARGE	Coulomb	C[†]
TEMPERATURE	Kelvin	K
AMOUNT	Mole	mol
INTENSITY	Candela	cd

PML will use the fundamental units of measure similar to those developed by the International Bureau of Weights and Measures (Le Système International d'Unités – SI) in conjunction with others such as the National Institute of Standards and Technology (NIST) in the United States. The seven quantities selected as the basis of the PML measurement specification are shown in Table 1. These are identical to the SI units except Coulombs were selected as a base unit instead of current, which is a measure of charge flow.

Given a set of base units, most physical properties will not be independent. Speed, for example, is the ratio of length to time. PML will define measurement based on multiples or ratios of the fundamental units.

PML defines a **measure** element with an option **label** attribute, which is a human-readable tag. Seven other optional attributes, **m**, **kg**, **s**, **C**, **K**, **mol** and **cd**, define the exponent of the unit of measure. If an exponent attribute is omitted it is assumed to be zero. An optional **accuracy** attribute indicates the number of significant digits in the measurements. Thus the PML definition of **measure** is given by:

```
<measure label="String" m=Integer kg=Integer
    s=Integer q=Integer K=Integer mol=Integer cd=Integer
    accuracy= Integer>
    float
</measure>
```

For example, atmospheric pressure – 101.325kPa, 76 cm of Hg, 101,325 m^{-1} kg s^{-2}, 760 millimeters of mercury, 14.7 lb/in^2, 1013.25mb, 29.92 inches of mercury, 34 ft of water, 101,325 Pascals – is

```
<measure label="Pressure" m=-1 kg=1 s=-2>
    101325.0
</measure>
```

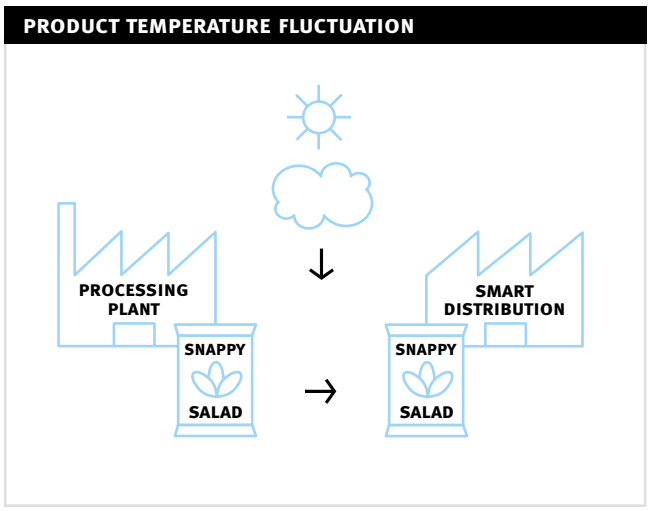
and there is no other way to describe it.

Example

Suppose that due to weather conditions – cool at the processing plant and warm at the distribution center – together with a faulty cooling system, there is a slight rise in temperature within the truck as shown in Figure 5.

Listing 3 shows how this information can be contained in a series of **data** elements. Note that the first temperature measurement shows the units in absolute values of Kelvin. This particular measurement corresponds to 40° F or 4.44° C. Note that in the last **data** element, the temperature went up to 51° F, which might be out of range. This series of data elements can occur in many places within the PML file, include the **trace** element. In this case, you would be able to associate temperature measurements with the owner information at the time the data was collected.

Figure 5. Due to the ambient temperature, some of the salad experience a slight rise in temperature from the processing plant to the distribution center.



Listing 5.
The temperature variation of the reusable plastic container (RPC) is captured, transmitted and stored in the PML file.

```
<data>
    <date>33935521256B</date>
    <measure label="Temperature" K=1 accuracy=5>
        277.59
    </measure>
</data>

<data>
    <date>39376845715</date>
    <measure label="Temperature" K=1 accuracy=5>
        283.71
    </measure>
</data>

<data>
    <date>394021207770</date>
    <measure label="Temperature" K=1 accuracy=5>
        288.70
    </measure>
</data>
```

5. CONCLUSION

In this paper we present some of the core components of the Physical Markup Language – particularly those that specify configuration, location, time and measurement. These are only some of the core components that will be included in the initial specification. Other elements will include specification for classification, attributed data (cost, price, discount, etc.), presentation information (text, icons, displays, Web sites, etc), physical properties (geometry, composition, mass and inertia) and others.

We should note this is not a specification, but indicates the line of development toward an initial specification. It is likely the names, structure and 20 will change in the initial specification. It is unlikely, however, the specification will lose functionality from that presented here. In other words, the initial specification will allow richer and more detailed object descriptions.

Furthermore, the nomenclature may change. Although fully descriptive elements and attributes aid understanding, they must balance with data storage and transmission efficiency. It is likely more efficient naming methods will be adopted.

6. REFERENCES

1. **Brock, D. L, "Intelligent Infrastructure – A Method for Networking Physical Objects,"**
   Presentation, MIT Smart World Conference, Apr 2000.
   http://auto-id.mit.edu/whatsnew/download/DB_Smart_World.pdf

2. **"The Networked Physical World – Proposal for Engineering the Next Generation of Computing, Commerce and Automatic-Identification,"**
   Auto-ID White Paper, WH-001, Dec 2000.
   http://auto-id.mit.edu/pdf/MIT-.AUTOID-WH-001.pdf

3. **Brock, D. L, "The Electronic Product Code – A Naming Scheme for Physical Objects,"**
   Auto-ID White Paper, WH-002, Jan 2001.
   http://auto-id.mit.edu/pdf/MIT-AUTOID-WH-002.pdf.

4. **Brock, D. L, "The Physical Markup Language – A Universal Language for Physical Objects,"**
   Auto-ID White Paper, WH-003, Feb 2001.
   http://auto-id.mit.edu/pdf/MIT-AUTOID-WH-003.pdf.

5. **Radio Frequency Identification (RFID) summary from the AIM Global Network**
   (http://www.aimglobal.org).
   http://www.aimglobal.org/technologies/rfid/

6. **Motorola BiStatix Technology**
   http://www.motorola.com/GSS/SSTG/smartcard/3_0_bst_home.htm
   http://www.motorola.com/GSS/SSTG/smartcard/white_papers/BiStatix_Whitepaper.pdf

7. **The Object Naming Service (ONS) summary from the MIT Auto-ID Laboratory**
   http://auto-id.mit.edu/research/naming.html.

8. **St. Laurent, Simon, "XML™: A Primer, 2nd Edition,"**
   MIS Press, New York, 1999.

9. **The Extensible Markup Language (XML) Specification World Wide Web Consortium**
   http://www.w3.org/XML.

10. **The Document Type Definition (DTD) Specification World Wide Web Consortium**
    http://www.w3.org/XML.

11. **The Resource Description Framework (RDF) Specification World Wide Web Consortium**
    http://www.w3.org/RDF.

12. **The XML Schema Specification World Wide Web Consortium**
    http://www.w3.org/XML/Schema.

13. **The Electronic Data Interchange, American National Standard X12 syntax.**
    http://www.x12.org/

14. **The Electronic Data Interchange, United Nations/Electronic Data Interchange for Administration, Commerce and Transport (UN/EDIFACT)**
    http://www.unece.org/trade/untdid/.

15. **The Electronic Business XML (ebXML) Specification**
http://www.ebXML.org/.
United Nations Centre for Trade Facilitation and Electronic Business (UN/CEFACT)
http://www.unece.org/cefact/
The 20 for the Advancement of Structure Information Standards (OASIS)
http://www.oasis-open.org/.

16. **The Universal Description, Discovery and Integration (UDDI) Initiative**
http://www.uddi.org.

17. **Simple Object Access Protocol (SOAP)**
http://www.w3c.org/TR/SOAP/.

18. **Logsdon, T., Understanding the Navstar: GPS, GIS and IVHS,**
Van Nostrand Reinhold,
New York, 1995.

19. **International System of Units (SI) from the National Institute of Standards and Technology (NIST)**
http://www.nist.gov
http://www.nist.gov/cuu/Units/units.html.