WHITE PAPER

A Manager's Introduction to Adobe eXtensible Metadata Platform, The Adobe XML Metadata Framework

Adobe

The eXtensible Metadata Platform

is Adobe's description format for Network Publishing. This new framework is an electronic labeling system for files and components of files, designed so that

Computers

can read

and understand

the labels,

and populate the information automatically into the right fields in databases, respond to software agents, or interface to intelligent manufacturing lines, just to name a few of the implications. Goodbye to hot folders; hello to true workflow automation.

Among its competitors, only Adobe has the size and scope to amortize an investment of this scale. It is making the investment in the interests of its users. Adobe applications are about content creation, and this investment in XMP will enable its users to mobilize their content across the boundaries of different uses and different systems. XMP allows the content creator to broadcast the data outwards. It allows author-centric production. In short, XMP will give Adobe users a value-ad supercharge.

The people who work in the eye of the Web hurricane with World Wide Web Consortium (W3C)—Tim Berners-Lee, for example, who invented Web in 1989—capture this matter of meaning and understanding with in their concept of the "Semantic Web". As they put it, "the Semantic Web is a vision: the idea of having data on the Web defined and linked in a way that it can be used by machines not just for display purposes, but for automation, integration and reuse of data across various applications."

They have spent countless hours over many years building up the infrastructure of new Web technology standards to serve as the superstructure for the transition from today's "Dumb Web" to tomorrow's Semantic Web. With XMP, Adobe takes the leadership position among all software companies in implementing these new standards.

Implementing the new standards for metadata integration is a large technical and financial undertaking. Adobe is implementing the core technology in all Adobe applications, it is making it public and extensible, to users and developers of content creation applications, content management systems, database publishing systems, web-integrated production systems and document repositories.

Introduction ii

The bean data analogy 3

A gentle technical introduction to XMP 9

XMP components 10

XMP framework 10

XMP schemas 12

XMP development kit 14

XMP extensibility 16

Summary and conclusion 16

Brief annotated bibliography 18

The Bean Data Analogy

We humans take labels for granted. But suppose the cans on the local grocery store shelf looked like this. We would have no clue about the contents. Could be dog food as well as green peas. It would make shopping (and eventual consumption) more of an adventure, to say the least. With no labels to convey understanding about the contents, the grocery business would hardly work at all.

So we put labels on the cans (and boxes and bags, and, increasingly, right on pieces of fruit), and we humans read the labels and understand their meaning and make informed choices. The labels make it possible for the cans to cross the boundaries between the shelf and our cart, our cart and check-out, the grocery bag and our pantry shelf, and eventually between our pantry shelf and our cooking pots.

Let's examine one of these labels in detail, the blue and red one on the upper right hand corner of the stack, the can we immediately identify as the black bean can because that is the first data item that we comprehend as we scan the stack. *(Figure 1)*

There is a lot of bean data on this label. As the table below shows, there are thirteen separate bean data elements. The first eleven are the manufacturer's optional information, for humans to read. The twelfth is a bar code for a laser scanner to read. The thirteenth element is a structured table with content and format specified by Federal Government regulation.

In summary, what has evolved over time in the grocery business is a system of labels that convey the meaning necessary to make the business work. The labels are composed of elements that have a definite and sometimes (as in the case of the "Nutrition Facts") rigid structure. Each element has a category, a data type, and a value. Most of these elements are human-readable, and make immediate sense to humans. For one of these, the tracking number, we need a separate key to understand. Another, the bar code, is machine-readable and encodes meaning in a standard form.

4

ELEMENT NUMBER	CATEGORY OF INFORMATION	VALUE OF CATEGORY IN THIS INSTANCE (What appears on the label)	DATA TYPE
1	The maker:	Trader Joe’s	String
2	The contents:	Black Beans	String
3	A notion of distinctive food value:	A low fat food	String
4	A second notation of distinctive food value:	An excellent source of dietary fiber	String
5	Directions for finding nutritional information:	See side panel for nutritional information	String
6	A notation of weight, in English and metric units:	New Wt. 15 oz (415g)	Formatted numbers
7	A marketing narrative	Trader Joe’s Black Beans have a rich, hearty taste and soft texture. They are wonderful in soups and stews, with rice, and in salads with colorful vegetables and Southwestern or Caribbean flavors. Black beans have gained in popularity due to their high dietary fiber and protein content. They are a cholesterol-free and low fat food.	Long string
8	A declaration of wholesomeness:	No preservatives, no artificial colors, no artificial flavors	String
9	A list of ingredients:	black beans, water, salt, calcium chloride	List separated by commas
10	The ID of distributor and seller:	Dist.& Sold Exclusively by Trader Joe’s, So. Pasadena, CA 91031	String
11	A tracking code, in Roman	0009 6362	Integer
12	Same tracking code in bar-code-readable format		Bit map
13	The nutritional facts, in standard order and format:	**Nutritional Facts** Serving Size 1/2 cup (130g) Servings per container about 3 Amount per serving **Calories** 130 — Fat Cal 5 % Daily Value **Total Fat** 0.5g — 0% Saturated Fat 0g — 0% **Cholesterol** 0mg — 0% **Sodium** 260mg — 11% **Total Carbohydrates** 22g — 7% Dietary Fiber 5g — 22% Sugars 0g Protein 10g — 20% Vitamin A 0% ϒ Vitamin C 0% Calcium 4% ϒ Iron 10% • Percent Daily Values are based on a 2,000 calorie diet	Structured table

figure 1 ***Bean Can Label***

Bean-flow analysis

Applying this label is not a trivial expense, but it is more than justified by the need for understanding if the can is going to successfully negotiate the multiple"bean-flow" system interfaces from production line to cooking pot.

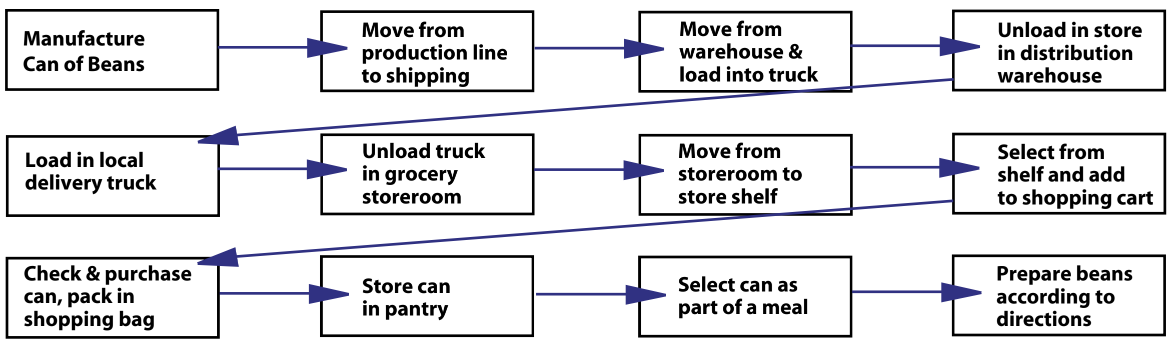

figure 2 Bean-flow from manufacture to consumption

From bean data to metadata

Generally speaking, the situation with media content is even more complicated than with groceries, and, from the perspective of computers, rather than humans, the labels are woefully deficient or non-existent.

The more typical degree of complexity is illustrated in Figure 3, which shows the simple passage of a news photo from source through to print publication, web publication, and newspaper archive.

The same block of data, a digital photo, passes through four different content creation applications, two different digital asset management systems, and across two different boundaries between design and delivery technologies. At the end of the process, the photo it has been archived as itself, as part of a Web page, as part of an InDesign layout document, and as part of a PDF document.

The archiving process is a particular challenge. In the emerging standard for an archive structure, the original digital format is held in a file store, referenced by a pointer in a record in a database that is managed by some "middle ware" responsible for storing and retrieving content in response to request from both humans and computers.

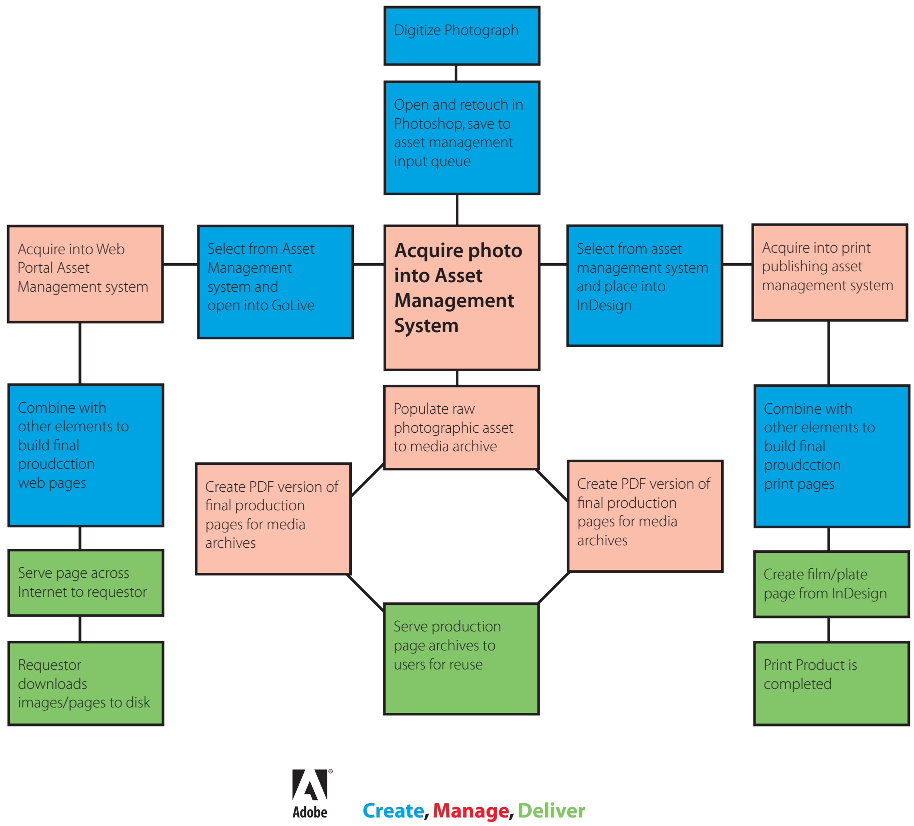

figure 3 ***Sample workflow for a digital photograph***

The latest versions of Adobe InDesign, InCopy, Photoshop, GoLive and Illustrator provide the basis for ubiquitous implementation of this structure. With WebDAV connectivity built-in, the Adobe suite of applications function as clients to a Web server, just the way a browser such as Netscape functions directly as a client of a Web server. This means that by selecting the appropriate menu item in InDesign, for example, a user can request a file be delivered from a file store via the Web server or save a file in the same manner.

What is needed to manage this type of interaction efficiently (i.e., automatically, without the intervention of pricey human thinker/database-entry-clerks) is a machine-readable label of elements based on a system of categories, and—to facilitate verification routines—specified data types.

A third kind of labeling requirement is based on the evolution of Adobe's own set of content authoring applications. Adobe is the leader in a paradigm shift from what might be called "electronic paste-up" to a process of assembling interactive components. In the former case, the way a page is assembled is to create the components in photo or illustration applications, translate them into some intercommunication format, such as PDF, and then place them on the page. This is an exact electronic analog of the way paste-up worked before computers. In contrast, with the breakthrough application Adobe InDesign, the original Adobe Photoshop or Adobe Illustrator files can be imported directly into the layout.

Using XMP-enabled publishing technology, this page could be put together in an InDesign layout, using placed files from Photoshop, Illustrator, or InCopy.

figure 4 ***Typical Newspaper Page***

The applications could be used interactively:

How does InDesign know what application to open, and what precise file to open into that application? The answer, once again, is a machine-readable label. When the original file from, say, Photoshop, is placed into InDesign, the elements in the label Photoshop file are incorporated into the InDesign file, ready to provide the meaning that guides the interoperability of applications, or meaning that can be used to set values of fields in databases.

These examples— workflow interfaces, database support, and application interoperability — make it clear that Adobe had to implement a system for machine-readable labels, a semantic system, if only to make the content creation applications work well internally. There were three approaches to consider, and the features of each are arrayed in this table.

Option	Accessible to Adobe developers?	Completely under Adobe's control?	Leverage work of Web developers?	Benefit from decentralization?
1 Proprietary	NO	YES	NO	NO
2 Semi-proprietary	YES	YES	NO	NO
3 Open, W3C standards	YES	NO	YES	YES

The advantage of the proprietary system is the ability to exclude any application or solution vendor from doing any extensions. This narrow and exclusionary philosophy has never been part of the Adobe business philosophy.

The advantage of the semi-proprietary system is that developers "in the family" can have access, but Adobe still controls. The disadvantage is that Adobe has to "reinvent the wheel" and compete with World Wide Web Consortium (W3C) standards.

Perhaps what tips the balance towards the semantic system developed by the W3C is the leverage it provides. Adobe, its developer partners and its users benefit from hundreds of man-years development, testing and refinement by the experts in the World Wide Web Consortium. The technology is complete, not beta. There are already a variety of toolsets for extension developers available on all platforms.

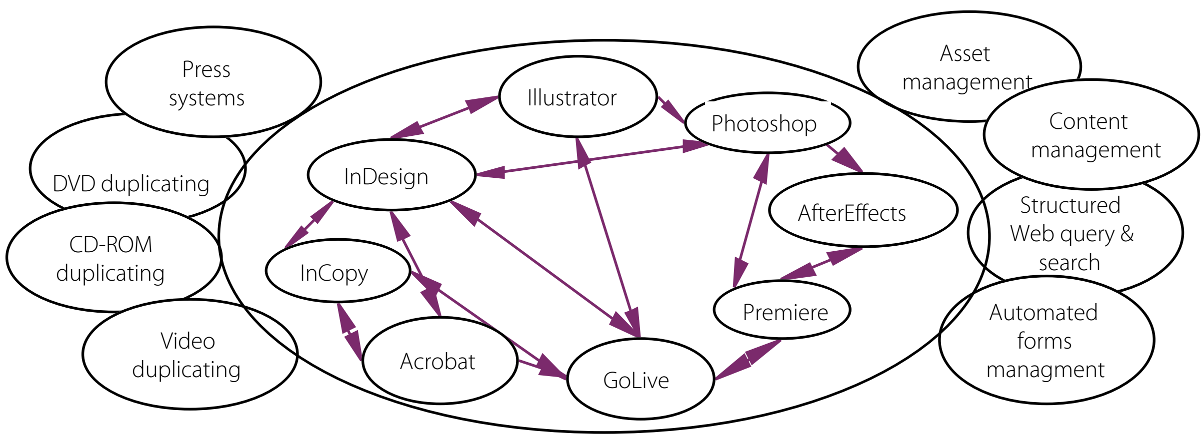

figure 5 ***Author-centric production***

Finally, the whole environment— the internal interactions of Adobe applications as well as the external interactions with solutions build on Adobe applications—can benefit from the innovations that stream in from all over the world in the decentralized Web environment. Control is sacrificed for creativity, and Adobe developers and users are assured that the labeling system they will be working with enjoys the fullest measure of features and functionality.

Therefore, the Adobe eXtensible Metadata Platform, "XMP", was built on the W3C standards.

A gentle technical introduction to XMP

eXtensible Metadata Platform XMP is a framework for adding machine-readable labels, or semantic content to application files, databases and content repositories.

History Every operating system vendor and every application vendor has always had some way of labeling their files. Historically, these have been rather primitive.

The full scope of the metadata, " the data about the data"[2], has been kept proprietary, only fully available to the employees and associates of the originating vendor. That metadata has always been too sparse to support efficient interaction. Even where the metadata may have been comprehensive and accessible, the utility of the labeling systems has been minimized. Absent a single world standard, there was not sufficient incentive for different applications and systems vendors to build in support for common labels.

The lack of comprehensive metadata emerged as an acute problem for the evolution of the World Wide Web. Humans can get a lot out of a file with HTML tags, because the browser converts the data to ordinary language that people understand. However, for a machine that does not understand a human language, the HTML does not add comprehension. A file marked up only with HTML tags is no more informative about its content to a machine than a can without a label is informative to supermarket patrons.

Starting in the mid-nineties, a group of key Web technologists set out to remedy this situation, to create a second generation Web technology that could be used to make Web pages and other resources machine comprehensible. They developed a system of two new "languages." The first, the Resource Definition Framework, or RDF, is for structuring the labels. For machine reading, RDF is implemented in XML expressions. The Semantic Web is not XML. XML is a language used to build the Semantic Web.

XMP components

The Adobe Metadata Framework is one of first major, comprehensive implementations of RDF. The elements of the Adobe XMP platform are:

RDF Framework or expressing metadata from multiple schemas – **XMP Framework**
Schemas used to describe properties, contained in namespaces – **XMP schemas**
Method for embedding XML fragments in binary streams – **XMP Packet Technology**
Support for third party interface and extensions to XMP – **XMP SDK**

The XMP framework

provides the means by which metadata from multiple master and sub-components can be combined. Just as Trader Joe's Black Beans label is an array of printed elements, a XMP label is a sequence of items of metadata, or metadata elements. Within applications, the "document" notion corresponds closely to a file created in any of the desktop applications when a user selects New from the File Menu.

Whether for print, web, or video, a document is typically composed of sub-documents. The complete content item is created by assembling sub-documents (chapters in a book, spreads in a magazine, clips in a video).

The important point is that XMP framework respects this operational reality: Where a document is assembled from sub-documents, each containing a metadata label, the sub-document label is preserved in the containing document composed of sub-documents.

The notion of a sub-document is quite flexible. Sub-document status can be assigned to a simple block of information—a photograph for example—as well as a complex one, like the

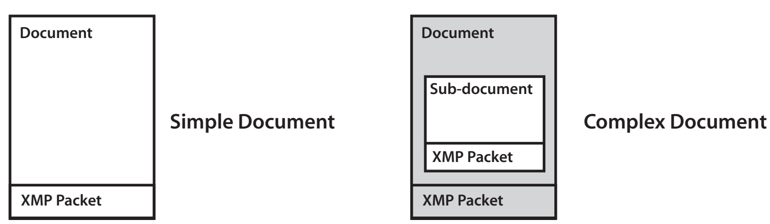

figure 7 ***Simple vs. complex document metadata embedding***

[2] "Metadata" is technical term that involves a slight twist of an original meaning. The dictionary meaning of the prefix meta is "Denoting a nature of a higher order or more fundamental kind, as metalanguage, metatheory." (The New Shorter Oxford English Dictionary). In information technology, however, metadata means data about data, or data used to describe data. Similarly, metalanguage (XML, for instance) is most often used to mean a language about language or a language used to build a language.

photograph along with its caption and credit. Simple sub-documents can be nested within larger, complex sub-documents.

Consider a corporate report. In some contexts it is sufficient to have just one simple label, that includes, say, the title, document type, author, and date. In other contexts, it may be appropriate to include as well labels for each section of the report. In still others, it may be appropriate to label the different charts inside each of the chapters.

With the XMP framework, each of these is possible. Indeed the label system can go on an on. If there is something that can be identified, a label can be attached to it.[3]

While there is a tendency to think of the labeling complexity in terms of subdivisions and components of documents, it is also important to realize that the labeling is particularly important in the organization of a publishing workflow. So, for example, in some editorial groups, the task of captioning is assigned to a particular person or team. As a result, it might be useful for workflow tracking purposes to label the caption data element, even though the label might require many more bytes of data than the caption itself.

It is also important that in some contexts the label might need a label. The label on the corporate report, for example, might include indication of review and approval by the legal department, and the note about the legal department might include a label recording the personnel in the legal department involved in the approval.

In practice, the way a document will be labeled will be usually be determined by the organization that creates the document, and the labeling standards to which they adhere. Newspaper publishers will use specific systems for labeling the contents of each day's paper. Hardware manufacturers will build a labeling system for the maintenance manual for each product. In some cases, standards bodies will determine the labeling architecture for a class of documents, and many of these standards already exist. One example is the labeling architecture devised by the IPTC for news photos.

The well tempered label

RDF, the Resource Description Framework is a formal way of rendering into a label the commonsensical way of describing something. One might say of our can, "This can contains black beans; it was made by Trader Joe's; it has tracking number 00096362." Each of these sentences consists of a subject, predicate, and object:

Formally, anything that is a resource can have a label, and a resource is anything with a Uniform Resource Identifier (URI), and a URI is a "compact string of characters for identifying an abstract or physical resource." (Berners-Lee, et. all, "URI Generic Syntax", RFC 2396, August 1998). The URI is something that identifies a resource the way a Social Security Number identifies a US citizen.

Subject	Predicate	Object
This can	contains	black beans
It	was made by	Trader Joes
It	has tracking number	00096362

Similarly, basic rule for the RDF system is that a label consists of a number of properties, all of them providing a triplet of information:

A subject, = Resource being labeled with the property
A predicate = attribute of the resource
An object = value of the attribute

One can configure a label by identifying the resource or subject of the label, then listing attribute value pairs,

In bullet form	In XML for RDF
Label about this can	<rdf:Description about=can>
• Contents: black beans	<Contents>black beans</Contents>
• Maker: Trader Joe's	<Maker>Trader Joe's</Maker>
• Tracking number: 00096362	<TrackingNumber>00096362</TrackingNumber>
	</rdf:Description>

XMP schemas

The RDF rules specify the composition of label into a sequence of XML statements of structured as a triple of data called resource, property, value or alternatively called subject, predicate, object. The schemas expressed with RDF define the vocabularies used in the labels. The schemas are the collections of attribute types, corresponding value types, and in some cases the specific alternative values that can be specified. The schemas specialize the general labeling system to one that is a labeling system appropriate for a particular domain of knowledge.

For the Adobe Metadata Framework, Adobe has created the "Standard XMP schemas." These schemas are a starting point, but critical to the value of the XMP framework is the ability to include *any* schema, provided it is defined according to the specification. Domain-specific schemas, like IPTC or NewsML, for example, can be described within XMP Packets.

The XMP initiative implements these schemas as required by the standard, namely, in a machine-readable XML format and also in a human-readable table format. As an example, consider the documentation for the XMP dynamic media schema named "Video." The human-readable table presents the name of the property, the value type for the property, and a description of the property.

Property	Value Type	Description
BitRate	Integer	Bits per second
Dimensions	Dimensions	Size of playback view rectangle
Interleaved	Boolean	If true, NTSC fields, otherwise frames
NaturalRate	Real number	Fields/Frames per second

The final constituents of the XMP schemas are purely human readable explanations of the more opaque properties. The table below is an example of such an explanation, in this case for the VersionID property in the XMP Multi-media Schema.

Property	Description
XMPMM:VersionID	The document version identifier for the target resource Each version of a document gets a new identifier. Usually these values are simply incrementing integers 1,2,3... etc. Media management systems may have other conventions or support branching which requires a more complex scheme. The Version identifier should be kept short. This property should be used primarily by a media management system. Applications with sufficient interfaces to detect user intent of creating new versions, can assign new version identifiers at appropriate times, but should be careful to avoid conflicts with media management systems.

In summary, it is the creation and documentation of the XMP schemas that Adobe implements the semantics or meaning of the metadata system.

The XMP Packet

For a Web page, inserting these labels is a matter of putting in the XML expressions. But Adobe will be inserting these into files created by applications Illustrator, InDesign and other applications. Adobe has done this by specifying a binary structure called a XMP Packet. This packet contains the label and exists as part of the application file. Schematically the XMP Packet looks like this:

figure 8 ***Simplified XMP Packet structure***

The XMP packet structure is intended to make it easy for third party software to find the labels by scanning through application files. The XMP format includes a number of rules for implementing XMP packets to insure that the scanners are not confused by ambiguity or burdened with excess operations.

XMP Development Kit (SDK)

Shared as an open-source license, XMP is available for integration into any system or application. Different environments will require different types of integration, but one thing is clear. Because XMP is based on W3C standards, developers will not get caught in the ugly trap of integrating integrations — a common problem in modern publishing system architecture.

Questions that have to be answered relate to application usage — what are the standard adjustments to metadata when a new file is created, an old file opened, on a save, or save as? Another question addresses the interaction of media management systems with applications.

A third requirement addresses the complex issue of document embedding, such as a page embedded into a PDF file by Acrobat, or a Photoshop picture embedded into an InDesign file. These issues are critical in newspaper editorial systems, for example, where in a conventional workflow a story is written in an external word processor, saved to a database, flowed to an InDesign page, copy edited in InCopy, and archived with the edits as a new version in the database. Without proper integration, the metadata gets jumbled and the meaning is lost in complex exchanges such as this.

One of the reasons the XMP initiative is the first big XML/RDF metadata project for publishing is the very significant cost of creating the entire development infrastructure. The programming tools and toolkits are an example of costly items that do not make sense if a company has only a single publishing application. In Adobe's case the ability to reuse the implementation code in a broad array of applications justifies the cost of the internal developer tools.

Adobe will be providing basic tools for the external community as well, including XMP libraries, SDK's and other resources. It is anticipated that as the technology matures, suites of developer tools will come to market, especially in the area of interfacing XMP to existing enterprise database and media management environments.

While the XMP schemas have been designed to respond to kinds of metadata requirements encountered most frequently in media publishing, Adobe has designed the system to respond to particular requirements of different applications vendors and especially to special requirements that may be part of asset management and content management systems.

Perhaps the most common source of special requirements comes from the adaptability of XMP to database systems. A solutions developer can use build a correspondence between a database record and the metadata in the XMP packet. And then use the values of the XMP label in a particular file to set the values of fields in a data base record that tracks the file.

Often, the asset-management system vendor or the system user will have a need for fields that do not correspond to any properties in the Standard XMP schemas, creating a need to flow the special fields and values into the metadata. Adobe has provided for this through the XMP extensibility features.

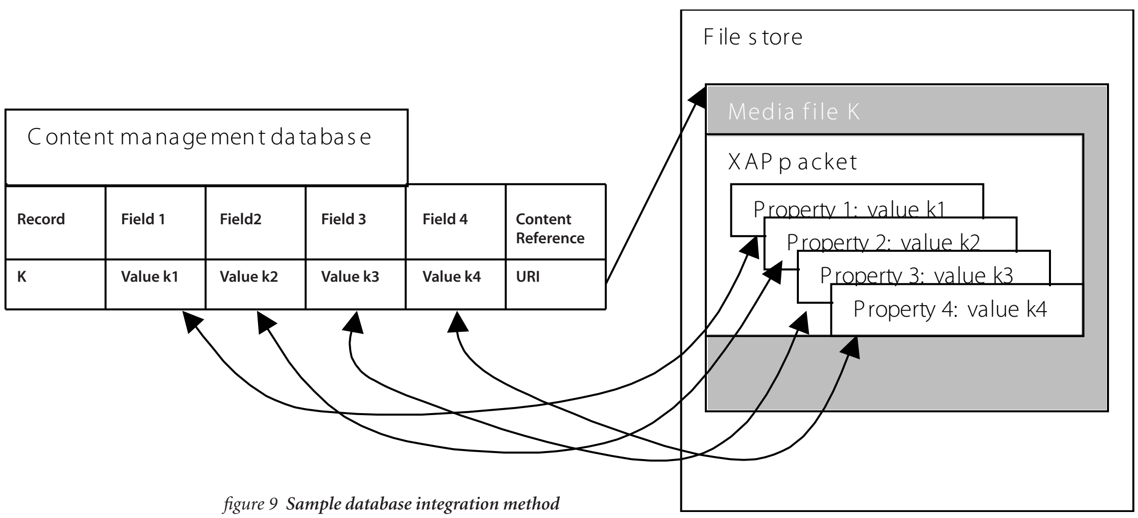

figure 9 ***Sample database integration method***

XMP extensibility

allows a system vendor or user to create a custom XMP Schema, following the XMP specification. This requires both a machine-readable XML representation and a human-readable table that is made available to all XMP developers. Once the new schema is in place, the custom property:value pairs can be added to the data in the XMP packet, and they will be respected by all XMP processing routines just as if they were property:value pairs corresponding to standard schemas.

In some cases the same concepts are used but the names for the properties are different. XMP provides for this by supporting aliasing from a non-standard name to a name in one of the standard schemas.

Summary and Conclusion

Whether in Web, print, or video publishing, content professionals have come to rely on Adobe Systems for creative tools with unparalleled creative scope and sophistication. XMP is another step in that direction:

- Increasingly, the creative activity takes place within an integrated workflow that assembles complete publications from a myriad of components. Downstream of assembly, a content composed for newspaper broadsheet will be re-purposed for distribution via server to a Web browser on a PC, to a PDA, a cell phone, collected into an archive, and perhaps even pressed as a CD.
- A printing plant wants to set its Web agents to work looking up-stream at the incoming flow of jobs, and automate the delivery of paper and ink, schedule the presses and the pressmen, and, increasingly, automate the process of plate production, color key adjustment and make-ready.
- A businesswoman wants her agents to comb through the financial pages of the great papers of the world and alert her whenever there is news that can affect her industry.

In all of these cases, something new is necessary to make the system work under machine control in a decentralized fashion: namely, a means to embed semantic content in files and file components, implemented according to open standards, accessible to any software the follows the specification, a system like XMP.

XMP is a huge milestone along the road towards a widespread implementation of this new generation of functionality. Adobe is in the leadership role. XMP is the first major implementation of the ideas behind the Semantic Web, fully compliant with the specification and procedures developed by the World Wide Web Consortium. Adobe has invested in this technology to automate the interaction of its own applications, and to permit solutions developers to create interfaces to content management, software agents, and manufacturing systems.

XMP raises the bar for integration in the content space.

Brief annotated bibliography

1. Readers are urged to begin here:

The Semantic Web, Tim Berners-Lee, James Hendler and Ora Lassila,

Scientific American, May 2001 issue, ***http://www.sciam.com/0501issue/0105berners.lee.html***

This article is a general introduction to the title subject by the three individuals who have perhaps done the most to make it happen. They describe themselves as "...individually and collectively obsessed with the potential of Semantic Web technology." The article shows how the technology can serve personal informational needs, especially for personal agents. It also gives insight into the way the Semantic Web can enable the interface of manufacturing systems and inventory systems, as well as the interaction of Web-enabled systems within the home.

2. Other resources

Semantic Web Activity: Resource Description Framework (RDF), W3C ***http://www.W3C.org/***

This site has an overview of the RDF development timetable and a collection of links to other key documents.

Resource Description Framework (RDF) Model and Syntax Specification, W3C ***http://www.W3C.org/***

This is a long technical document that serves as the formal spec for RDF.

Guide to the Semantic Web, Diffuse, ***http://www.diffuse.org/semantic-web.html***

This is a comprehensive introduction to RDF, RDF Schemas, and the technologies that can be built on this foundation for agent technology and other automated processing. The first few pages are general, and they are followed by a technical introduction.

Adobe Systems Incorporated • 345 Park Avenue, San Jose, CA 95110-2704 USA • *www.adobe.com*

BCXXXX XX/00