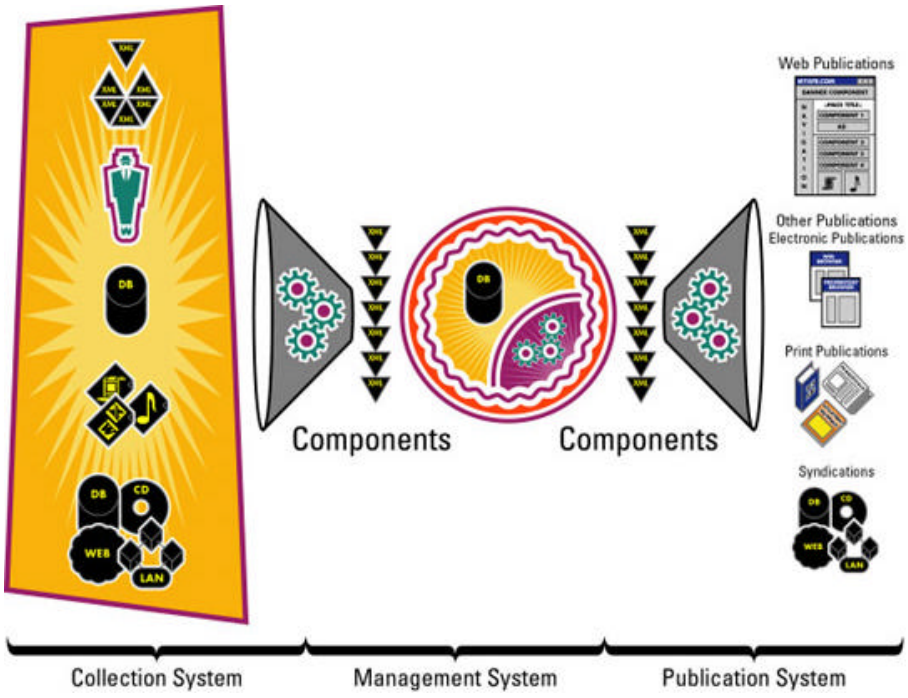
Working with Metadata

A CM Domain White Paper

By Bob Boiko



This white paper is produced from the Content Management Domain which features the full text of the book "Content Management Bible," by Bob Boiko. Owners of the book may access the CM Domain at www.metatorial.com.

Table of Contents

Metadata is the small snippets of information (or data) that you attach to content so that you can more easily catalog, store, and retrieve it. A coherent system of metadata draws diverse classes of content into a coherent scheme in which content components relate to each other as well as to the collection, management, and publishing systems that you devise.

In this white paper I dive into the concept of metadata, discussing its meaning, types, and uses.

What Is Metadata?

More than anything else, the metadata system behind a CMS is what defines the system. The set of names and relationships that a metatorial framework contains are the skeleton on which you

hang the content. Without this structure, the content is as formless and flaccid as a body without bones.

What does meta mean?

The word *meta* itself has no meaning except (according to the Oxford English Dictionary) as a column that marks the boundary of a circus. In more general usage, *meta* is a *prefix*. It modifies the meaning of the words that it precedes.

Merriam-Webster OnLine (at www.m-w.com) states that, as a prefix, *meta* means the following:

- Occurring later than or in succession to.
- Situated behind or beyond.
- Change or transformation.
- Later or more highly organized or specialized form of.
- More comprehensive or transcending

All these definitions give depth to the idea of meta that I have in mind in this white paper. Meta-things come after the things themselves to add context and organization to them. They go beyond the things themselves to a higher level of abstraction, where you see the things in a wider and different light.

I first became enamored by the concept of meta while reading Douglas Hofstadter's book, *Godel, Escher, Bach: An Eternal Golden Braid*. This most wonderful work uses meta as one of its most central and subtle threads. In the following dialog, Achilles (a character Hofstadter borrowed from the Greek philosopher Zeno) meets GOD who turns out to be a never-ending hierarchy of genies::

> *Achilles: Tell me, Mr. Genie — what is a meta-wish?*
>
> *Genie: It's simply a wish about wishes. I am not allowed to grant meta-wishes. It's only in my purview to grant plain ordinary wishes, such as wishing for ten bottles of beer, to have Helen of Troy on a blanket, or to have an all-expense-paid weekend for two at the Copacabana. You know — simple things like that. But meta-wishes I cannot grant, GOD won't permit me to.*

You must read the book to find out exactly who GOD is and get Hofstadter's deep and wide view of metaphenomena. For the present, I need take only one word out of his work — *about*.

To my mind, the best way to understand the prefix *meta* is to substitute the word *about*. The word *metadialog*, for example, is a dialog about a dialog. A *metatorial* (not my usage of this word, but another common one) is an editorial opinion about editorial opinions. *Metadata* is data about data.

The word *about* captures the large majority of the dictionary definitions of meta. If you're talking about a thing, you must be coming after the thing, and you must be speaking at a transcending or higher level of organization than that of the thing itself. Most important, by talking about the thing, you're changing its meaning. My recent favorite, which I use on every new kid I meet, is a statement that is its own metastatement: "Everything that I say is a lie!"

Note

You may be familiar with the word *meta* from HTML where you have a <META> tag. This tag is now a catch-all for any information that people want to put in it, but its intended use is quite consistent with its name. You're supposed to use the <META> tag to list information *about* the HTML file.

What does metadata mean?

Just saying that metadata is data about data isn't much of a description. The term is in wide enough usage these days to have a lot more behind it.

You can get a fair feeling for how people use the word by looking at the following few sentences from the home page of the Metadata Coalition at www.mdcinfo.com/:

> *"The Coalition allies software vendors and users with a common purpose of driving forward the definition, implementation, and ongoing evolution of a metadata interchange format standard and its support mechanisms. The need for such standards arises as metadata, or the information about the enterprise data emerges as a critical element in effective data management. Different tools, including data warehousing, distributed client/server computing, databases (relational, OLAP, OLTP...), integrated enterprise-wide applications, etc.... must be able to cooperate and make use of metadata generated by each other."*

This excerpt, one among many that you can find on the Web, shows the main characteristics of metadata as people most commonly use the term, and the following list describes those characteristics:

- **Sharing:** Metadata provides the capability to share data across applications. By employing data that describes the use and meaning of the data that it surrounds, one system can interpret and translate the data that it receives from another. In a content management context, metadata enables publications that each need a somewhat different form of the same data to draw from a common repository.
- **Standards:** Metadata is a set of standards that groups agree to for information definitions. Standards, which are the basis of any kind of data sharing, bring the possibility of large-scale efficiencies in information interchange among groups that don't even know one another. In the content management context, the standards may be mostly internal today, but they serve the same purpose. Standards ensure that others can automatically reuse the efforts of one person or group if they all follow the same standards.
- **A focus on databases:** The main reason today for an interest in metadata is the sharing and standards behind "standard" database applications. Data warehousing and interapplication data transfer are huge concerns for organizations with an enormous amount of data trapped in databases and other files that no one can interpret except by using the application that created them.
- **An awareness of the wider world:** Today, metadata is mostly used in data systems. A vague but growing understanding exists, however, that metadata isn't just for data. The previous quote from the Metadata Coalition mentions "integrated enterprise-wide applications." A CMS is an integrated enterprise-wide application. Metadata is critical, not only so that a CMS can integrate with other enterprise data sources, but so that the CMS itself can unify and make the best automated use of the information and functionality that it manages.

  Note

  By the way, metadata, apparently, is a registered trademark of the Metadata Company, headquartered in Long Beach, California (and at www.metadata.com/ on the Web).

Merging this information with the definition of the prefix *meta* can give you a deeper understanding of what metadata is.

First, metadata is what you need other than the data itself to understand and use that data. Metadata acts as the instructions that come with the data. In addition, metadata is what isn't there if you look at the data itself. Metadata exists in addition to or after the data. It adds context and a

wider interpretation to the data. Consider, for example, the following piece of content that has no metadata:

> *Cinder Riley is the anti-Cinderella. She wants nothing to do with fame and riches and longs to run away with a pauper...*

Without any other information surrounding this content, knowing what to make of it or what you may use it for is hard. By adding some metadata, however, its meaning and use becomes clear, as the following table shows.

Metadata element	Value
Author	Claire Taylor
Publication date	January 1968
Source	Plays Magazine
Reviewer	Bernard Lewis
Audience	Children, primary school teachers, and librarians
Title	The Adventures of Cinder Riley
Review	Cinder Riley is the anti-Cinderella. She wants nothing to do with fame and riches and longs to run away with a pauper...

Now you can tell that the content is a review of a play called "The Adventures of Cinder Riley." All that extra information isn't the content; it surrounds the content and tells you *about* it. In addition, the chunk of text alone is hard to make use of. But add metadata and the chunk becomes part of a wider system of shared attributes that help to store, locate, and present it in the context of the other content that you may have.

The metadata isn't the content. I make this claim because it exists apart from the content. Contrast the preceding table with the following representation of the same basic information:

In January 1968, Claire Taylor published "The Adventures of Cinder Riley" in *Plays Magazine*. Bernard Lewis provided this review of the play, which is of most interest to children, primary school teachers, and librarians. Cinder Riley is the anti-Cinderella. She wants nothing to do with fame and riches and longs to run away with a pauper...

In the table form, *Claire Taylor* is a piece of metadata. In the paragraph form, it's not. What makes *Claire Taylor* metadata not only is the fact that it's descriptive of some other content, but also that it's separate from that content. As a content manager, you choose what information to separate out and make into metadata and what remains part of the content. This distinction is a practical, not a philosophical one. If you use the words *Claire Taylor* as metadata, you must somehow name and separate it so that a computer can find it and use it to categorize the content that it surrounds.

As a final point, notice that nothing is exclusive about putting metadata in a table that extracts it from the content that it surrounds. The following paragraph, for example, uses XML tags to distinguish metadata from the content it's embedded in:

In < PublicationDate>January 1968</ PublicationDate>, <Author>Claire Taylor</Author> published <Title>The Adventures of Cinder Riley</Title> in <Source>Plays Magazine </Source>. <Reviewer>Bernard Lewis<Reviewer> provided this review of the play, which is of most interest to <Audience>children, primary school teachers, and librarians</Audience>. <Review>Cinder Riley is the anti-Cinderella. She wants nothing to do with fame and riches and longs to run away with a pauper...</Review>

What does metatorial mean?

Over the years during which I've been building electronic publication systems, I've hit again and again on the problem of metadata. At first, I didn't think of metadata as a problem unto itself. Rather, I confronted related issues such as, "How do I know what content is ready for an audience to see?" or, more widely, "How do I pack a bunch of data in with the content, knowing that this data isn't going to appear but that I need it to manage the production and display of the content?" Over time, I began to standardize the ways that I dealt with this "bunch of data" so that it became easier and easier to find, parse, and manipulate. For some time, I used a "dot" system, where each line of the file that I needed to process began with a period and the name of the data field. The systems of standardized tagging that I was working with made isolating metadata from the information it surrounded and doing what I needed to do with it (usually find it and load it into database fields) easy. At about the same time (I would guess about 1996), the term *metadata* became fashionable in the electronic publications arena as a way to describe "data about data."

As is apt to happen, over time I began to notice some patterns, as the following list describes:

- **Metadata consistently fell into a few distinct categories.** These categories include *navigational* (TOC, index, cross reference, and browse), *management* (author, create date, last edit, and so on.), *content type* (what I now call *components*), *internal structure* (title, abstract, body, and so on), and *inclusion* (add media here, add a banner here, add standard text here, and so on.).
- **Without a rigorous consistency and careful attention, metadata becomes useless.** What good is sometimes including a chunk of standard text and sometimes typing it in from scratch? How useful is a TOC if every submitter has a different notion of how organize content and where it belongs? (You can guarantee that every submitter's own content always comes out on top!) Someone or, better, some system is necessary to ensure that people handle metadata thoroughly and consistently.
- **A different set of skills is necessary to deal with this metadata.** Authors, who are, generally, subject-matter experts and not experts at the system for managing their expertise, are often unable to add this extra data. One very fundamental piece of metadata, for example, concerns where, in the overall outline of content, a particular piece of information belongs? In larger systems, authors who have no problem creating content often don't know enough to decide where to put it. The problem is worse for cross-references. Authors rarely have the wherewithal to discover what else others are submitting and exactly how to relate it

to what they submit. Overall, you need a type of person who's a cross between an editor, a librarian, and a database administrator to do a good job creating and maintaining metadata.

- **To do metadata well requires a lot of human energy.** Simply writing some scripts and pulling out what's easy to find isn't sufficient. A perfect example of this idea is keywording. I've seen many systems fail miserably at indexing because people thought that all they needed to do was create a program to find and mark every example of a small set of words and phrases.

With all these patterns in my head and with clients becoming ever larger and more demanding about the quality of their publications, I became creative. What's needed, I reasoned, is something like an editorial framework for metadata. It needed the right kind of leader (such as an editor), an overall metadata framework that everyone can buy into and apply, and a set of rigorous metadata guidelines (such as an editorial guide) and a metadata process that everyone must follow (such as an editorial process). Thus the *metator*, the metatorial guide, and metatorial processing were born.

As much as I like coining words, I recognize that it isn't the particular names that you choose that matter. Rather, it's the fact that you name a thing at all that's important. The most important thing for me in coming to these names was that doing so gave me a reason and a way to begin thinking about metadata systems as separate entities. Metadata became, for me, not just a part of gathering and tagging content but a subject of concern and importance on its own and apart from any particular content management process. For me, this turn represented a critical change from doing content management systems to *understanding* them.

The narrow view of metadata

The definitions of meta and metadata that I describe in the preceding section clearly point to a wide-ranging and deep notion of metadata. A more narrow view of metadata is nonetheless the most well-understood and accepted version that I've encountered. As best I can tell, this notion arose from the document-management industry and now informs most of the uses of metadata in content management products and literature. The following is my admittedly ill-informed interpretation of how the term evolved to its current usage.

First came file properties that the operating system exposed and put a user interface around. File name, size, creation date, and file type were some of the first kinds of metadata available. Although they weren't called metadata, these file properties clearly were metadata. They weren't the files themselves but rather data about the files.

After the document-management industry came along, one of the things that product companies did was to extend these file properties to include all manner of other data with which you could tag a file to describe it. Over time, people began to apply the term *metadata* to the set of properties that they could assign to a file. Document-management systems used these properties to help store, manage, and retrieve files.

Today, people still most commonly use the term *metadata* to describe a set of management data that you can use to categorize and manage complete chunks of content. In a CMS, the chunks may not be files, but the concept is the same. Metadata is administrative data about the content that enables you to manage its use.

The wide view of metadata

The narrow view of metadata corresponds best to my notion of the management component elements. *Management elements* are the administrative data (such as author, create date, and status) that you attach to a component to help you keep track of it, find it, and know what to do next with it.

But aren't the body elements of a component metadata? And how about the tags within an element that tell you what a piece of text is or does? Why wouldn't you call that metadata? In addition, why can't you call your access structures metadata? Don't hierarchies and cross-references offer data about the content in your system?

In fact, my view of metadata goes far beyond administrative attributes to include anything in a CMS that's data about the content you're storing. To some extent, the difference between the narrow and wide views of metadata is a consequence of a general principle of metas: What is the thing itself and what is the meta-thing depends on your level of awareness and concern.

For someone concerned with entire files, who never cracks those files to see what's inside, the thing itself is a file. The metathing is data about the file. In a CMS, you're concerned with smaller content chunks (which I call component elements), and you're concerned with the parts that make up the chunk (the formatting and structural tagging that lie within an element). So a wider view of metadata assumes that you need to mark entities smaller than an entire file with metadata.

A wider view also assumes that you need to manage metadata at a level larger than a file. Someone whose last task is to retrieve a file generally has no need to capture metadata on anything larger than a file. In a CMS, you need to name and track a number of things larger than files. These things are, for example, publications and access structures that include components.

I'm not saying that the narrow view is wrong and the wide view is right but to take the view that gives you the most advantage in the task that you're trying to accomplish. In a CMS, you need to take a view of metadata that's wide enough to encompass all the levels of content that you must manage.

Metadata and content management

If content management is the art of naming information, metadata is the set of names. In other words, content management is all about metadata. I reproduce it here for your convenience (see Figure 1).

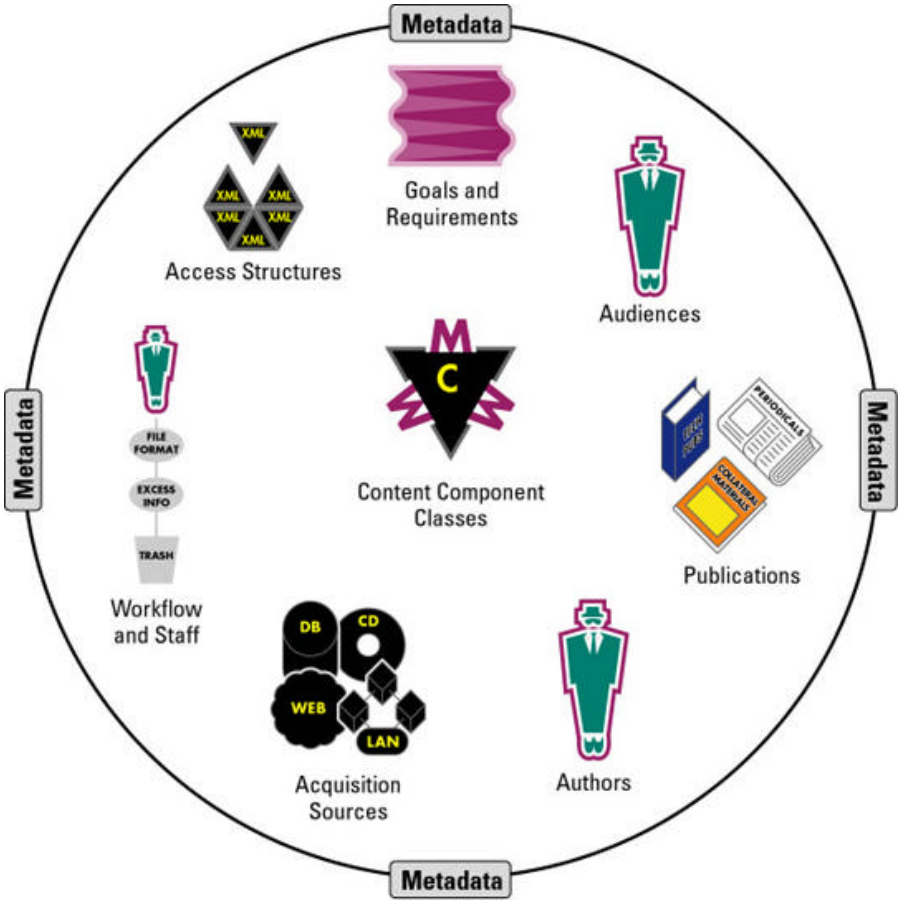


***Figure 1**: The wheel of content management*

Metadata forms the rim of the wheel. It holds together the entire system and ensures that it neither collapses in on itself nor expands out of control. Control is the essence of metadata in a CMS. If you name and account for everything, you're organized enough to exert complete control over your content.

Understanding the Types of Metadata

In the spirit of casting a broad metadata net, I offer the following list of the types of metadata you're likely to encounter in your travels through the domain of content management:

- **Structure metadata:** The ruling monarch of metadata. It precedes most other kinds of metadata by creating structural divisions in your content.
- **Format metadata:** Applies to any level of structure that you define and marks how you intend to render that structure.
- **Access metadata:** Organizes the structures that you create into hierarchies and other access structures.
- **Management metadata:** The data that you attach to structures to administer and track it.
- **Inclusion metadata:** Stands in for external content. It marks the place where the external content is to go.

One of the best ways that I can think of to get quickly immersed in the wide view of metadata is to look at some good markup. Markup languages (such as XML) are the major way that you apply metadata to content. The other major way that you apply metadata is in database rows and columns.

In the sections that follow, I discuss each of these types of metadata in more detail and give markup examples of how you may apply them to content.

Structure metadata

Structure metadata says, "You call this stuff..." It's the most basic of the kinds of metadata in the sense that, before you can say anything more about something, you must name that something. Structure metadata creates things by separating them out from their surroundings. In a CMS, structure metadata divides text all the way from defining character boundaries to dividing large collections of publications, as the following list details:

- **Characters:** The smallest structural unit that you're likely to need to notice. Because formatting metadata can apply to individual characters, you often can't ignore this smallest unit. Because one character is the smallest unit of text that a computer can store (you can't type half a character), you don't need a specific marker to show where one character begins and the other ends.
- **Words:** Collections of characters that you intend a reader to take as a unit. Spaces or punctuation set off words. You can say, although doing so stretches the definition a bit, that the spaces and punctuation are the metadata that tells you that a word is a word.
- **Paragraphs:** Collections of words that you mean for a reader to take as a unit. You mark paragraphs by using specific metadata to show their boundaries. (In HTML, for example, you use the <P> tag, while in flat text, you use a carriage return). Paragraphs are important to notice because you can apply format metadata at the level of a paragraph.
- **Elements:** Collections of characters, words, or paragraphs that you intend the reader to take as a unit (such as a title). Elements overlap with paragraphs and words in their range. You can have a paragraph composed of multiple elements, for example, as well as an element composed of multiple paragraphs. What distinguishes an element is that it's the smallest structure that you intend to access separately in your system. You mark elements by using specific metadata to show their boundaries (for example, database columns in relational databases and element tags in XML).
- **Components:** Collections of elements that you intend the user to take as a whole (such as a white paper). In a CMS, components are the structures that you intend to manage. Thus they're the structures to which you apply management and access metadata. You mark component boundaries by using specific metadata (for example, rows in relational databases and element tags in XML).
- **Nodes:** Collections of components that, after publication, you intend the reader to take as a unit. On Web sites, nodes are pages. In print materials, nodes are sections (headings, chapters, parts, and so on). Nodes put a standard set of surroundings around the components that they contain. In this sense, the node itself is metadata to the components. Nodes say, in effect, "You are to understand these components in the context of the surroundings in which you see them." On Web sites, file boundaries surround the node. (Each node is an HTML file.) In print, you're most likely to mark the node boundaries by using format metadata (heading styles and the like).
- **Publications:** Collections of nodes that you intend readers to take as a unit (a single department's intranet site, for example). On the Web, you set off publications from each other mostly by using graphic conventions and the internal navigation conventions of the site. (A site may have one or more publications on it). In print, you most often delineate publications

by using a file boundary. The publication is metadata to the nodes as the nodes are metadata to the components that they contain. The publication says, "Take these nodes in the context of the wider publication."

- **Publication groups:** Collections of publications that you intend the reader to take as a unit (the volumes in an encyclopedia, for example). Publication groups are set off on both the Web and in print by the formatting conventions and navigational structures that you provide for moving between publications in the group. Again, the group is metadata to the particular publications providing a wider context in which to interpret the meaning of the constituent publications.

The common threads between all these structural divisions are that each defines a unified whole that you can separate from its surroundings; each puts a wider context and another level of meaning around the ones that it contains; and you must somehow mark each division (even if only with formatting metadata or even graphical conventions).

Following is a simplified example of some XML that demonstrates many of the types of structure that I describe previously:

```
<COLLECTION>
   <PUB>
      <SECTION>
         <NODE>
            <HEADER>...</HEADER>
            <COMPONENTS>
               <COMPONENT>
                  <ELEMENT>
                     <PARA>
                        <COMPONENT>...</COMPONENT>
                     <PARA>
                  </ELEMENT>
               </COMPONENT>
            </COMPONENTS>
            <FOOTER>...</FOOTER>
         </NODE>
      </SECTION>
   </PUB>
</COLLECTION>
```

Even such a simple example, however, shows a lot about structure metadata. First, notice that no data appears in this example only metadata. The description is there, but I leave out the thing that it describes (for simplicity). Next, notice the nesting. Each wider set of metadata literally contains the ones within it. The interpretation of a component deepens as you reach the node, the section, the publication, and even the collection level. Thus all the higher levels are metadata of the component (or meta-metadata, meta-meta-metadata, and so on).

Finally, notice the overlap between components, elements, and paragraphs. In this example, an element contains a paragraph that contains another component. This situation isn't only okay; it's common. Consider the following more realistic XML example:

```
<ELEMENT>
   <PARA>
      This is my body text, and in it I'm embedding an image.
      <MEDIA ID="m1" URL="dabw.jpg">
         <SIZE>100,300</Size>
         <CAPTION>This is a separate component</CAPTION>
      </MEDIA>
   </PARA>
   Normal things seem strange if you really think about them!
</ELEMENT>
```

The <MEDIA> component (in this case, an image) I embed within the paragraph, which I embed within the element, which I embed within a component. If I render the <MEDIA> component in HTML, it turns into content and a piece of inclusion metadata, as follows:

```
<IMG SRC="dabw.jpg" WIDTH="100" HEIGHT="300">
<H6>This is a separate component<H6>
```

Format metadata

Format metadata says, "Here's how to render the stuff that I surround."

Format metadata can apply to any level of structure in your system. In many cases, the structural tags themselves are what you interpret and turn into platform-specific formatting metadata. Take, for example, the following structural XML metadata:

```
<SECTION LEVEL="1">Some Section</SECTION>
```

You can turn this example into format metadata for presentation on a Web page, as follows:

```
<H1>Some Section</H1>
```

Quite often, formatting metadata is "under the radar" of the CMS by residing inside an element. You may, for example, permit authors to include HTML codes for bold, italics, and underline in the Web forms that they use to input a component. The codes sit, unparsed in a database field, and make it onto a Web page without anyone ever noticing them. Of course, if the component's destiny is that of any other format besides HTML, you need a more robust approach.

Following is the XML structure example, enhanced to show a variety of formatting metadata:

```
<COLLECTION>
   <PUB DISPLAY="child">
      <SECTION>
         <NODE>
            <HEADER>...</HEADER>
            <COMPONENTS LAYOUT="table">
               <COMPONENT>
```

```
                    <ELEMENT TYPEFACE="Arial">
                       <PARA STYLE="body">
                       Some <FORMATTAG>text</FORMATTAG>
                          <COMPONENT>...</COMPONENT>
                       <PARA>
                    </ELEMENT>
                 </COMPONENT>
              </COMPONENTS>
              <FOOTER>...</FOOTER>
           </NODE>
        </SECTION>
     </PUB>
  </COLLECTION>
```

Although I've cheated a bit by putting some formatting in the content that normally goes in templates, the example still holds.

Here are some of the formatting concepts that the preceding XML illustrates:

- **The <PUB> tag** has the attribute DISPLAY as an addition to it. This attribute controls the display of the publication within the collection. It says, "Display the publication in a child frame of the collection." The display metadata value thus controls the rendering of the publication. This metadata is usually in a template and not a content structure.
- **The <COMPONENT> tag** has the attribute STYLE. If you render the paragraph, the formatting associated with the style "body" surrounds the entire contents of the PARA element. For each type of publication that you produce, you can associate different formatting codes with the same style name.
- **The <PARA> tag** has the attribute STYLE. If you render the paragraph, the formatting associated with the style "body" surrounds the entire contents of the PARA element. For each type of publication that you produce, you can associate different formatting codes with the same style name.
- **The word text** has the metadata <FORMATTAG> surrounding it. If you render the word, the formatting associated with the tag surrounds the word. For each type of publication that you produce, you can associate different formatting codes with the same tag.
- **Any tag** can have formatting associated with it even if it isn't specifically designated as format metadata. You may decide, for example, that headers on the Web are 12-point bold text. To do so, you treat the <HEADER> tag just as you do the <FORMATTAG> tag. If you render the header, you make sure that the software surrounds it with 10-point bold formatting codes.

  Note

  You may ask, in a tag such as <PUB DISPLAY="child"> is PUB the metadata, is DISPLAY the metadata, or is "child" the metadata? PUB and DISPLAY are the names of metadata. PUB is a Boolean metadata element (either on or off). DISPLAY is a list type of metadata element. Of the allowable values in that list, "child" is one of them.

Access metadata

Access metadata says, "Here is how this structure fits in with the rest." I call it access metadata because you most often use it to gain access to the content. You can, however, just as easily categorize it as structural metadata because it describes the logical structure of the content. Actually, calling it *structure meta-metadata* is more correct, because it describes the structure of the structures. Still, I prefer to differentiate it from structure metadata because you use different techniques to store and manage it. You can store access metadata within a component or outside it in a separate place. The types of access metadata correspond to the types of access structures: hierarchy, index, associations, and sequences.

Following is the XML structure example, this time enhanced to include some access metadata:

```
<COLLECTION>
   <PUB>
      <SECTION>
         <NODE KEYWORDS="rollup">
            <HEADER>...</HEADER>
            <COMPONENTS>
               <COMPONENT INDEX="term1,term2,term3">
                  <ELEMENT>
                     <PARA>
                     <COMPONENT>...</COMPONENT>
                        <LINK TARGET="C123">For more info,
see</LINK>
                     <PARA>
                  </ELEMENT>
               </COMPONENT>
            </COMPONENTS>
            <FOOTER>...</FOOTER>
         </NODE>
      </SECTION>
   </PUB>
</COLLECTION>
```

To show a hierarchy and sequence, you need nothing other than what I show here. By its nature, XML provides a default hierarchy and a default sequence. To show indexing, I add the following two new attributes:

- **The <COMPONENT> element** has an INDEX attribute where it lists the index terms for the component.
- **The <NODE> element** has a KEYWORDS attribute that specifies how the node is to treat index terms. In this case, the attribute says, "Roll up all the index terms from all the components that are on the node." On an HTML page, this command may result in the creation of a <META> tag at the top of the HTML file that lists all the index terms for search engines to find.

To show associations, I add a <LINK> tag. The tag includes a TARGET attribute that names the structure that it links to and some text that it uses in rendering the link. In a Web page, the result may look as follows:

```
For more information, <A HREF="C123.htm">click here</A>
```

In print, the link may look as follows after rendering it:

```
For more information, see "Links" in 5.
```

Access metadata is as often outside the content structure as it is inside. The way I that show index terms, for example, isn't the best way to do them. Instead of typing the terms into the component, you want to type the component into the terms as in the following example:

```
<INDEX>
   <TERM>
      <NAME>NOAA</NAME>
      <COMPONENTS>C123,C456,C789</COMPONENTS>
   </TERM>
</INDEX>
```

This approach is a better way to manage a large index. More to the present point, it places index metadata outside the content structure and instead references the structure.

Management metadata

My notion of management metadata is quite close to the common definition for metadata itself. Management metadata is there to help you keep track of and administer content. Following are some of the more common management metadata types:

- ID
- Title
- Author
- Create data
- Modify date
- Status
- Size
- Owner
- Publish date
- Expire date

Notice that management metadata isn't always only for management. Any of the types that I list here you can just as easily consider as content to publish as well as data to help manage the content to publish. That's fine. Notwithstanding the limitations of some CMS systems, whether or not you show the values of these metadata elements to your audience, their use to you is the same, to help you keep track of and administer your content.

Following is the XML structure example that I enhance to show some management metadata:

```
<COLLECTION>
   <PUB ID="p1">
```

```
            <SECTION ID="s1">
               <NODE ID="n1">
                  <HEADER>...</HEADER>
                  <COMPONENTS>
                     <COMPONENT ID="C123">
                        <TITLE></TITLE>
                        <ADMIN>
                           <OWNER>O234</OWNER>
                           <CREATE>9/23/01</CREATE>
                           <MODIFY>9/30/01</MODIFY>
                           <STATUS>Status1</STATUS>
                        </ADMIN>
                        <ELEMENT NAME="intro">
                           <PARA ID="p1">...</PARA>
                        </ELEMENT>
                     </COMPONENT>
                  </COMPONENTS>
                  <FOOTER...</FOOTER>
               </NODE>
            </SECTION>
         </PUB>
      </COLLECTION>
```

The management metadata in this example is fairly self explanatory. First, every significant entity gets a unique identifier so that you can always precisely locate it. Components get a set of new elements that specify the management metadata that you want the component to contain. For clarity, I make these new elements all XML tags. In real life, many would be XML attributes instead.

Inclusion metadata

Inclusion metadata says, "Put the following external entity here." It enables you to reference content that isn't physically in the content structure.

Recall the example that I present in the section "Structure metadata," earlier in this white paper that you use to embed one component in another:

```
      <ELEMENT>
         <PARA>
            This is my body text, and in it I'm embedding an image.
            <MEDIA ID="m1" URL="dabw.jpg">
               <SIZE>100,300</SIZE>
               <CAPTION>This is a separate component</CAPTION>
```

```
      </MEDIA>
   </PARA>
   Normal things seem strange if you really think about them!
</ELEMENT>
```

If you really intend to make the <MEDIA> element a separate component, you're better off not directly embedding it in another component by pointing to its URL but instead referencing it there based on its ID, as follows:

```
<ELEMENT>
   <PARA>
      This is my body text, and in it I'm embedding an image.
      <INCLUDE REFID="m1">
      Normal things seem strange if you really think about them!
   </PARA>
</ELEMENT>
```

The <INCLUDE> tag, on evaluation by the code that renders the page in HTML, finds component "m1" and puts it at that spot. In the meantime, component "m1" isn't stuck embedded in a particular place in the content structure; it's stored with the other "m" components where you can more easily find, manage, and include it in other places in the content structure.

A common use for inclusion metadata is in publications to reference media, as I show in the section "Structure metadata," Earlier in this white paper Here, for example, is how to reference an image in HTML:

```
<IMG SRC="dabw.jpg" WIDTH="100" HEIGHT="300">
<H6>This is a separate component<H6>
```

You may be tempted to use the following approach in your content structure as well:

```
<ELEMENT>
   <PARA>
      This is my body text, and in it I'm embedding an image.
      <IMG SRC="dabw.jpg" WIDTH="100" HEIGHT="300">
      <H6>This is a separate component<H6>
      Normal things seem strange if you really think about them!
   </PARA>
</ELEMENT>
```

Before you do, however, pause and consider the following issues:

- **This is HTML.** Are you sure that you can translate this publication-specific formatting metadata into whatever other type of formatting metadata you may need?
- **The image and its caption are locked in this location.** Do you need this image elsewhere? If so, you're stuck making multiple copies of the exact same lengthy version of the tags and text. By referencing an image component rather than literally typing it, you avoid the

redundancy. In addition, with a reference, you change all instances of the component by changing it in one place.

- **The reference may break.** As HTML embedded within a component element, this image may fall under the radar of your CMS. If "dabw.jpg" changes names or moves to a different place, most CMS products don't tell you that the reference is no longer valid. In addition, you must somehow remember that this image file is referenced in this element and make sure that it's published anywhere that the element it's part of is published. This task could prove a tall order.
- **You have no place to put other info that you may need.** Suppose that you wisely decide that your images need to carry a status that tells you how ready they are for viewing. If you represent them as components, the change is easy. Simply add a status element to the component. But if you represent your images as HTML references, where do you put the extra information?

Although just typing media references into your text may seem convenient, hidden costs could outweigh the time that you save up front.

Finally, consider the HTML notion of the INCLUDE file. This one uses Microsoft ASP syntax, as follows:

```
<! #INCLUDE FILE="Reusable.txt" >
```

HTML INCLUDEs are the poor man's way of simulating components and templates. INCLUDEs are essential in dynamic Web applications, where they enable you to "type once and include many." In a CMS, the "type once and include many" processes that are inherent in components and templates largely, but not entirely, replace the concept of the INCLUDE file.

Note

I'm a bit ambivalent about the inclusion metadata category, because it isn't as clearly metadata as the rest. You can surely say that the INCLUDEs are data about the content. But on the other hand, you can say that inclusion metadata is just a way to reference content that you don't want to type (or otherwise include) over and over again. I'm leaving the debate unresolved for now to enable me to cover the subject of inclusion in the most logical location for it.

Categorizing Metadata Fields

So far in this white paper I've looked at metadata from an XML perspective. In XML, all metadata boils down to elements and attributes. I now turn toward a more database-centric approach to metadata as I discuss how to categorize metadata not by the meaning it conveys, but by its allowed values.

Before discussing allowed values, I'm going to pause to mention briefly how the structure of a relational database itself represents metadata. Consider the simple database table shown in Figure 2.

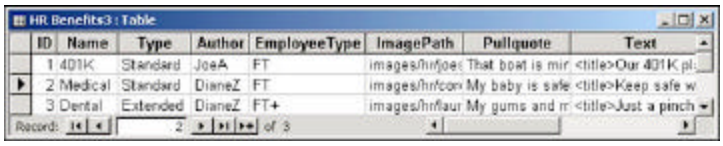


***Figure 2**: Rows are component boundaries and columns are element boundaries.*

The content is in the little boxes of the grid. The rest is metadata, as the following list describes:

- **The column names** are structural metadata that delineate element boundaries.

- **The rows** are structural metadata that delineate component boundaries.
- **Management metadata** is in the ID, Name, Type, Author, and EmployeeType columns.
- **Inclusion metadata** is in the ImagePath column.

The structure of the database itself as well as the ways that you assign values to database columns yields a lot of well-structured and strictly consistent metadata.

Continuing to look at metadata from a database perspective, you can say that metadata falls into a set of categories based on the values that the metadata element can have. The value of the metadata element Author, for example, may be "Karl Weyrauch." The value of the Create Date element may be "3 January, 2002." To make the transition from XML to databases complete, instead of metadata elements, I refer to them here *as metadata fields*.

Note

Those familiar with programming probably notice that I mix database and user-interface vocabulary with other words that come from neither discipline. I do so with full knowledge that I'm mixing metaphors to create a content management-centric view of metadata.

Metadata fields that one may find in a CMS fall into categories that include the following:

- **Free text**, where the value of the field can be any text string. Titles, for example, are of this variety because every title is different.
- **Constrained text**, where the value of the field is text, but the text must follow certain rules. A "BodyText" metadata field, for example, may allow any ASCII characters as well as the HTML codes for bold, italics, and underline. Or the field may put constraints on length, limiting the number of characters that can go in that field.
- **Pattern text**, where each value of the field must follow particular formation rules. Dates and times are of this variety because you can form them only in a set number of ways.
- **BLOB** (Binary Large Objects), where the value of the field is any binary data that's interpretable by some application. Image and other media fields are examples of BLOB fields.
- **Boolean**, where the value of the field is either true or false. You may, for example, create a field that you call "Public." If its value is true, people can view the component on the public Web site.
- **Closed lists**, where people can choose the value only from a predefined list. You may, for example, create a field that you call "Review Status," where the review status is one of only 10 possible values.
- **Open lists**, where someone can choose the value from a list or can add new entries to the list and then choose them. You may, for example, create a field that you call "Keywords," where the user can either select one of the keywords in the existing list or add a new one. If you use open lists, make sure that you trust users to add new items responsibly.
- **Unique ID**, where the value must differ from that of any other field. All components should have an ID field so that you can easily locate them in any database.
- **Outline lists**, where people choose the values from a hierarchical list. You may create a field that you call "Product," for example, where the user can select a product from an outline of product categories and products.
- **Compound fields** that combine one or more of the preceding types. You may, for example, create a "Link" field as a combination of an outline list and a reference field. To fill in the field, the user chooses from an outline of all the available components. What gets stored in the field is a reference to a component.

Notice that these categories span two realms: how you enter information into the field and what gets stored in the field. The compound field illustrates this concept well: One field type specifies the user interface (outline, for example), and one provides the storage. In this sense it's a very CMS-centric view of field value types. It focuses on what's important to the management system and its users. A more general (and, to you, less useful) approach is to take a data-centric view and focus on the data type of the field. (Is it text, integer, real number, BLOB, and so on?) Although the programmer who sets up your CMS database uses its standard data types, some additional coding is likely to prove necessary to combine and uniquely present options in the user interface that serve the metatorial needs of the users.

Introducing the Metatorial Framework

A CMS needs to automatically store, retrieve, and publish content. To do so, it needs a set of rules to ensure that the repository stays organized and that automatically generated publications are well-formed and unified. These rules are the *metatorial framework*. Just as editorial rules govern the presentation of the content, metatorial rules govern its management and accessibility. In static publications, you manage and access content by hand. In other words, the rules about how you manage and access the content are in someone's head. In dynamic publications, you must make these rules explicit in metadata and programming code so that the computer can use them to manage and access the content. Whereas an editorial framework provides rules for creating content, a metatorial framework provides rules for organizing, tagging, and targeting content.

The metatorial framework consists of the follows components:

- **The metadata system** that you create to organize everything that you know *about* the content that you need to manage and its use. This part of the framework you codify in the metatorial guide.
- **The meta-metadata system** that you devise that contains rules *about* how to organize, change, and extend your metadata system. You can also document this system in your metatorial guide.
- **The programming code** and other rules that you establish that dictate how you access and use metadata. You don't have any single place where you can gather all these rules. Rather, you express the rules in template logic, CMS configuration, collection tool configuration, and myriad other places where you must specify what happens to content based on how you tag it.

Another way to look at the metatorial framework is that it's the sum total of logical design. Although *logical design* is a better name for the process, the result is everything that you need to know *about* the content that you're managing. As does the logical design, a solid metatorial framework ensures that your design is sound and that users of the CMS are all contributing to a growing, useful base of knowledge.

On the other hand, with so many other deliverables in a CMS project, I see no reason to create a separate framework document. The metatorial guide and the logical design specifications are enough to document the framework that you devise.

Metatorial processing

Editorial processing ensures that your content obeys the rules that you construct for correctness, communication, and consistency. Metatorial processing ensures that metadata is complete and consistent.

Metatorial processing accomplishes the following tasks:

- Completeness of metadata.

✍✍ Consistency of metadata.

✍✍ Ensuring that you can manage the content.

✍✍ Ensuring that you can access the content.

✍✍ Ensuring that you can appropriately target the content.

Although either authors or a conversion system can do much of the tagging, they can't do it all. Authors may not possess the ability or desire to do all the tagging that you need, and an acquisition source may simply not have all the information that you expect in the target markup. Thus you may need additional processes that ensure that the tagging is complete for each content component that enters the system. Missing metadata is quite easy to find (a blank space appears where some value goes), so the problem here isn't to find the blanks, but to correctly fill them. Of course, with a well-written metatorial guide in hand, this task usually isn't too difficult, even if it's time-consuming.

Your system must ensure that metadata is consistent among all the content that flows into the system. You need to apply similar component classes to similar metadata. You must apply a particular metadata element (status, say) with an even hand across all components that it's a part of. In a sizable system with numerous contributors you can almost guarantee that you find wide variation in the ways that people interpret and apply even the most precisely stated tagging rules.

Complete and consistent metadata is a means to the real end of metatorial processing to make content manageable, accessible, and targetable.

Whenever most people speak about metadata they usually do so in the context of management. Data such as the author, create date, last edit, status, and so on that you attach to content not for publication purposes but rather to provide "handles" for accessing and doing something with it before you publish it (as well as to help you decide when to stop publishing it). Obviously, for the content to remain manageable, management metadata must be complete and consistent.

In addition to management metadata, other metadata makes the content accessible. This accessibility goes in two directions. Individual elements must be accessible within components. If you're to show summaries of each of your feature articles, for example, each Feature Article component must include a tagged summary. In the entire CMS, individual components must remain accessible by the metadata that positions them in hierarchies, indexes, cross references, and sequences. Needless to say, if you don't correctly tag components as separate entities (that is, you don't segment them correctly), you can't find them at all. So metatorial processing must result in complete and consistent access tagging.

Finally, metatorial processing must ensure that the tagging that you direct at targeting content for personalization is complete and consistent. Suppose, for example, that your logical design says that you target families differently than you do single adults. The simplest way to do so (simple conceptually, if not very practical) is to tag each component with a piece of metadata that you call *audience type*. You can imagine that the end system cant function well if some components are missing these tags (completeness) or if various contributors make various divergent decisions about which types of content goes which ways (inconsistency). In reality, targeting content is usually much more complex than this example, but the idea is the same. Whatever tagging you need to add to content to enable you to target it you must apply consistently and completely.

The metator

Much (and, I hope, most) content tagging is something that either your authors or conversion systems handle. If an author fills in a create date on a Web form, for example, she's tagging her content with a piece of metadata. Similarly, if a conversion program finds and converts an author name, it tags that author name for use by the CMS. Expecting your authors or converters to do all that tagging is a nice luxury, but realistically, it's pretty unreasonable to do so.

Authors don't always have the wherewithal to know how to tag their content. Yes, you can usually count on them to tag their names and other ready-to-hand information. But can (or, more important, should) you count on authors to know where in your overall outline of content to list their contributions? Depending on the size and complexity of your system and degree of involvement of your authors, you can't always count on them displaying much knowledge beyond the boundaries of their particular creations. As a general rule, authors should always be responsible for the internal structure and format tagging of their work. They should be responsible for relating their content to the rest of the system (the external structure of their work) only to the degree that they're experienced and involved in your metatorial framework. In addition, assuming that, because someone is a good author, she's can or wants to become involved in concerns that go beyond the boundaries of her contribution is a dangerous luxury. All in all, in a system of any size, you're better off planning for the likelihood that authors can't fully tag their own work by finding a metator.

Acquired sources simply may not contain all the metadata that you need to capture. Or the information may be present in the source, but so poorly marked that it requires a person to review and tag it by hand. In either case, it's the responsibility of the metator to ensure that acquired content is fully tagged.

Most basically, a *metator* is a person who does the metatorial processing that authors and conversion systems can't do. She may do so directly by adding XML or other tags to the content, or she may do so indirectly, by choosing from lists and typing into input areas that later result in the tagging other organizing of the content (for example, putting it into the correct database table).

In a small system, you can get away without appointing someone as an official metator. But in a system with dozens of authors and sources and a large variety of component classes, the job of metator becomes mandatory to keep the metadata system complete and consistent. What a metator does, in this regard, isn't dissimilar to what an editor does in her realm. She reviews, adds, and modifies not content but metadata.

In addition to filling in missing metadata, the metator must ensure that the metadata that's already present is consistent with the metatorial guide. Authors and conversion systems are apt to make poor choices if they face any ambiguity at all about how to use a tag. This statement isn't a pejorative. Laziness notwithstanding, squeezing all the vagueness out of any set of rules is impossible. (That's why lawyers exist.) Maintaining a central point of decision about a rules system becomes mandatory after the system reaches moderate complexity and more than just a few people are using it.

Whether or not the job title is metator, the task of looking after the metadata health of a CMS is crucial if you want to trust that the content in the system shows up when you expect it to.

The metatorial guide

You can think of a metatorial guide as the master catalog of your content. For every kind of content you manage, it fully describes its structure and construction. From another angle, you can say that a metatorial guide is all your system's meta-metadata. It doesn't describe the metadata itself; it's everything *about* the metadata. From yet another, more practical angle, you can say that the metatorial guide is the place to go to find out what a component consists of and how to fill in all its elements. However you choose to describe it, a metatorial guide, more than any other document, is at the center of your CMS. Metadata is the one place where all the parts of a CMS meet, and the metatorial guide is where you describe the metadata.

Tip

The metatorial guide needs to remain accessible for anyone to use it. If you can maintain it in only one form, make it a set of Web pages with one page per element. That's the most flexible form for

including it as online Help in your content collection forms as well as making it a standalone publication.

Meta-metadata rules

Meta-metadata rules tell you how to formulate and change metadata. They're not the rules for filling in metadata; they're the rules for changing how you fill in metadata.

You need to consider and document the rules, because as soon as you begin using the system, it's certain to need to change. If this change happens in an orderly way, the metadata system has the best chance of staying ordered. If it happens ad hoc, you can bet that the tight system you constructed is sure to begin to unravel.

Tip

Of course, as you begin to change, you must reconsider the rules of change, so you also need to construct the rules for the rules of change. But then those rules change, too, so you must also consider the rules for the rules for the rules of change. But those rules change, too, so... It's not a true meta if it can't go on forever!

In all seriousness, taking the time to figure out the following issues ahead of time is well worth the effort:

- **Control:** Who's in charge of deciding when you need to change a metadata element and when you need to add a new one? Elsewhere, I suggest a CMS steering committee. Is that what's right for you?
- **Lumping and splitting:** How do you decide when to combine two metadata elements into one (*lumping*) or when to create two elements from one (*splitting*). What methods do you use for these procedures? The effort behind splitting especially may prove enormous, so planing ahead is worth your while.
- **Top down or bottom up:** In general, do you want what you find in the content to generate metadata (*bottom up*), or do you want your team to generate metadata and impose it on the content that arrives (*top down*)? In other words, do you prefer a metadata system that expands on its own and risks disorganization (bottom up) or one that remains very organized but doesn't discriminate well between similar sorts of content (top down)?
- **Entry:** In general, what sorts of elements do you expect authors to fill in and what sorts require a metator?
- **Freeform or constrained:** How do you decide which metadata to keep open-ended (where you can add whatever you want) and which to close down (where you restrict what anyone can add)?
- **Optional or mandatory:** When do you make metadata optional and when do you make it mandatory?

The answers to these questions aren't yes or no, either one or the other. The right answers are all "Some of both." In addition, you need to think hard about a policy so that you can maintain an organized metatorial framework but stay prepared to examine each metadata element individually to craft the best approach for it.

Note

To say "meta-metadata rules" is a bit redundant. Rules are, by their nature, metadata. To become a metadata rule, a statement must talk *about* the metadata that it seeks to control. I use the phrase anyway, just to underscore the point.

Metadata and collection

A metatorial guide is an invaluable resource for keeping content collection on track, efficient, and pain free. You have a winner if your guide can provide quick answers to any question such as, "What do I type here?" More specifically, the guide should use the results of your logical design to address the following concerns:

- **Component classes:** What component classes exist? What is the name of each one and what's its source (or sources)?
- **Component elements:** What component elements does each component class consist of? What are their names and where do they originate?
- **Element types:** What kind of field is each element (free text, pattern text, open list, closed list, and so on)?
- **Element values:** For each element, what are the allowed values?
- **Usage rules:** How do you know what value to choose or type for each field?
- **Responsibilities:** Which elements are contributors responsible to fill in? Which do other staffers fill in? Which are automatically filled in?
- **Change rules:** For each component class or element, which of the meta-metadata rules that I cite in the preceding section are relevant and need describing to the contributing public?

Notice the switch here from the term *metadata element* to *component element*. You shouldn't expect contributors to know what metadata is. They do, however, need to know what a component is if they're to create them. In all likelihood, the word that contributors respond to best is "field." What they see isn't a component element nor a metadata element, but a field on a Web-based form that they must fill in.

Metadata and management

You collection staff uses the metatorial guide daily. Management staff uses it less frequently, but it should still prove a valuable resource if it addresses the following issues:

- **Review:** What are the review and quality-control processes for the elements of each component class?
- **Retirement:** What tells you that a component is ready for expiration, archiving, or deletion?
- **Creation Rates:** How many of each component class do you expect to add each week?
- **Growth:** Which component classes continue to grow indefinitely and which have a maximum number of components?

Much of this information you code directly into the configuration of the CMS. A component may display both a Retirement Type and a Retirement Date element, for example, that specify what to do with it. You can set up the CMS so that, as the retirement date arrives, a workflow trigger fires and sends an e-mail message to the administrator telling her what to do with the component (as the Retirement Type element specifies). Thus the administrator doesn't need to consult the metatorial guide to decide what to do. On the other hand, you probably have no place in the system to see all the management rules and metadata laid out and organized. If an administrator wants to see the big picture, the metatorial guide is the place to go.

Metadata and publishing

Publishers look at a metatorial guide even less frequently than administrators do. Still, as buried as metadata rules may be in the management system, they're even more buried in the publishing

system and templates. To help provide publishers a clear picture of how the publishing system is using metadata, the metatorial guide can address the following concerns:

- **Components:** What components does each publication use? Which do more than one publication use?
- **Templates:** For each publication template, which components does it use?
- **Selection:** What component elements do you use to select components into each template? (Do you select components because their Date element, for example, displays a date that's less than three weeks old?)
- **Personalization elements:** What component elements and what user profile elements do you use in the personalization process in each template?

You code all these rules directly into publication templates. The component elements that you use to select components into a template, for example, you type into the queries of the template. This approach is just fine if the system is working as planned, but if you need to find and fix a bug or make a change to the templating system, you need something more easily navigated than the code in each template. At these times, the overall picture that the metatorial guide provides can really help you. It can help you see inconsistencies (such as why a component appears fine in one template and not in others) and opportunities (for example, the needs of two templates are so close that they can just share code) that you may never see, let alone notice, if your metadata system is implicit in the publication system and templates.

Localizing Metadata

You need to localize metadata values that you publish to the world beyond your team just as you do any other content. If you intend to publish a create date for article components, for example, you must adopt the local date formatting conventions as you display it.

For management metadata that the end user never sees and for the names of the metadata elements themselves, avoid the duplication of effort and the system complexity that localization involves. You're better off without it. If you do decide to avoid localizing metadata, make sure that the following conditions are true:

- **The elements that you skip are pure management.** Make sure that no one down the line is likely to decide to start showing the end user any elements that you're not localizing.
- **You choose unambiguous words.** You must make the meaning totally clear in the metadata element names and metadata lists that you create. To save yourself work, understand that, if you use colloquial words, you're making things harder for the people who aren't native speakers of the language you're using. You can help a lot by using standard words (no jargon) and full words (no abbreviations) in the way that you name metadata elements and list values. If you don't do so as a favor to your international staff, do it to save yourself the extra effort of correcting their tagging mistakes.
- **You take extra care in your metatorial guide to explain the names and metadata values.** Consider localizing portions of the guide if you can't localize the entire system.

If you choose to localize element names and management metadata values, you may run into significant difficulties. Few systems make changing element names easy. Even providing an alternative list of values by region can prove a big task and can require extensive customization.

Summary

Metadata is data about data. More specifically, it displays the following characteristics:

- You can divide metadata into structure, format, access, management, and inclusion categories.

- In a CMS, metadata can have a variety of formats that vary from free text to references to components. You use a correspondingly wide range of user interface elements for metadata entry.
- To provide the organization and structure to turn the metadata that you manage into a working system, you must construct a metatorial framework.
- Just as an editorial framework has as its product an editorial guide, the metatorial framework results in a metatorial guide. The metatorial guide establishes the guidelines that staff members use to divide and tag all the content that crosses their desks. Metators and others use the metatorial guide to do metatorial processing on the content that you manage.
- You may or may not choose to localize management metadata, but you must localize any of the rest of the metadata that you capture that you may display to your local audiences.