Chapter 6

Technology Standards: Leveling the Playing Field

Digital Rights Management is an emerging technology market. At this writing, there are no dominant vendors in the DRM space — no standard choices to make when you're in the market for solutions, à la Microsoft for PC software or Oracle for databases. Eventually, some leading technologies will emerge. The question is whether they will be proprietary technologies, controlled by one or a small number of vendors, or based on open standards.

In the olden days of computerdom, single vendors (such as IBM) created large, monolithic information system architectures that customers had to adopt in their entireties. Since then, the industry has been moving more toward a "network economy" in which technology paradigms result from synergies among several individual components. Those synergies tend to spread farthest, and the resulting markets expand most rapidly, if the right standards are in place on which to base all the components. A standard is *open* if anyone who wants to use it can get access to it in its entirety, if it is controlled by a publicly accessible authoring process rather than by a single vendor, and if it is complete and unambiguous enough to serve as the basis for implementations.

The Internet industry is the ultimate example of a successful technology paradigm mostly based on open standards. The two primary standards, HTML and HTTP, form the basis of lots of different products, including Web servers, Web browsers, Web page authoring tools, site traffic analysis packages, and so on. Open standards promote markets in which multiple vendors put out products that are competitive yet compatible. When this is done right, customers end up with products that perform better and are more likely to satisfy their requirements. Contrast this with the PC industry, which is controlled by proprietary standards for microprocessors (Intel) and operating system software (Microsoft). Many people feel that PCs would be faster and more reliable if they were based on open standards.

The Role of Open Standards in DRM

For the Digital Rights Management market to take off, it must also settle on a set of technology standards that apply to the types of components discussed in Chapter 5: authentication, player formats, encryption, and so on. Most of the players in the DRM industry would prefer open standards. Certainly publishers would prefer to have a choice of vendors to work with to manage and serve up content, and consumers would similarly like to choose the best way of viewing or playing that content.

The only entities who may not favor open standards are deep-pocketed vendors who have the potential to monopolize the market and technology startups with dreams of becoming the monopolists (as well as the investors who back both).

There are also legal reasons why open standards are a good idea. As discussed in this chapter, some open standards have resulted from publishers, or their equivalents in related industries, getting together in trade association working groups and defining standards. It would be very efficient for such a group to convene, put out a request for proposals (RFP) to vendors, and have vendors bid on the right to be chosen as the preferred vendor for the industry. The vendor could build a solution (or offer one already built), and that would be the end of it. Yet antitrust law makes this illegal; it's analogous to price fixing in an industry such as petroleum or air travel. To pass legal muster, a standard must be *procompetitive,* meaning that it promotes competition among vendors rather than prevents it.

Types of standards relevant to DRM

Which aspects of the technology should DRM vendors standardize? Two principles are relevant here. First, because virtually all activity in the DRM world now takes place over the Internet in one form or another, DRM-related standards should work in conjunction with existing Internet standards. Second, standards should cover those aspects of content providers' businesses on which they will want to standardize; standards should not cover aspects that constitute a competitive advantage for individual content providers because they will want to keep those elements to themselves.

We will examine which aspects of content providers' businesses are proprietary and which are standardizable. Refer to the DRM reference architecture from Chapter 5. The primary categories of technology from the architecture are

- ♦ Components, such as content packagers and DRM controllers
- ♦ Protocols, or methods of communication between components.
- ♦ File formats
- ♦ Metadata, or information about content.
- ♦ Encryption and watermarking schemes

Technology components such as content packages and DRM controllers are not good candidates for standardization because publishers need a competitive market for these things. File formats are not good candidates both because there are already a few well-established standard formats (HTML, PDF, RTF, GIF, TIFF, JPEG, MPEG-1, MPEG-2, and so on) and because most decisions about file format standards are made well outside the purview of the DRM field.

Metadata is the most promising area for standardization. Protocols, which depend heavily on metadata, are also good candidates.

It would also be nice to standardize on encryption and watermarking schemes. As we see in Chapter 7, there are de facto standards for watermarking of images and audio.

Encryption is another matter. The U.S. government has advanced encryption standards such as DES and AES (see Chapter 5), but the cryptographic community tends to shy away from them for several reasons. First, some in that community don't trust the government to promote a standard algorithm that is truly strong (that is, impervious to cracking by government agencies such as the NSA). Second, cryptographers constantly tinker with algorithms and invent new ones that are allegedly more secure, and some of these have patents on them (or patents pending). Furthermore, different encryption algorithms are designed for different purposes. The result has been a profusion of encryption algorithms that is confusing and only understood by experts in the field.

You may expect there to be some open source standards for these technologies that work "well enough" and that everyone could adopt. Certainly the open source community has produced some popular and effective encryption-related standards, such as PGP (Pretty Good Privacy) for securing e-mail messages. However, copy protection is slightly different; it's something that the open source community frowns on in general. In particular, some in the open source community have been openly hostile to some of the standards discussed in this chapter that depend on strong encryption, such as the Secure Digital Music Initiative (SDMI).

Because metadata is a highly promising area for standardization, we will look at it more closely. Recall the types of metadata common to most content packages, as described in the DRM reference architecture in Chapter 5: identification, discovery, and rights.

Identification

Content identification is the most basic element of any DRM system. Just as SKU (stock keeping unit) numbers are necessary in the world of retail, each content item must be uniquely identifiable. Various segments of the content industries have invented their own identification schemes, most of which are tied to particular publication types — a veritable alphabet soup of identification schemes. Here are just a few examples:

- **ISBN (International Standard Book Number):** used for hard-copy books and related products, such as spoken-word editions
- **ISSN (International Standard Serial Number):** used for hard-copy serials, such as magazines and journals
- **LOC (Library of Congress) number:** used for books published in the United States
- **ISWC (International Standard Musical Work Code):** an emerging standard for musical works

None of these identification schemes was ever intended to be used for online content. The Internet, of course, uses uniform resource locators (URLs). URLs are very widely used, of course, but as discussed in the section "The Digital Object Identifier (DOI)," they have limitations when used to identify pieces of intellectual property. Another type of identifier that was developed with the online world in mind is the Publisher Item Identifier (PII), which has been fairly well established among scientific publishers such as the American Institute of Physics and the American Chemical Society, but is being phased out in favor of the more comprehensive DOI.

The Internet Engineering Task Force (IETF) has also been considering constructs called the uniform resource name (URN) and persistent uniform resource locator (PURL), which are both similar to the DOI. Work on the URN has been going on for several years, but at this writing it has not become an official Internet standard.

Therefore, the DOI has the highest potential as an open standard for content identification in DRM.

Discovery metadata

The most essential type of metadata beyond identification is *discovery metadata*, also called *descriptive metadata,* meaning metadata that enables the content to be found within a collection of content, such as a library or the Internet, if the unique identifier isn't known. For published works, discovery metadata would be likely to include bibliographic information, such as the title, author, publisher, publication date, and number of pages. It could also include a set of keywords that describe the book's subject matter.

It has been difficult to get publishers to agree on metadata sets because they can be used for so many different purposes. But one standard for bibliographic metadata has been gaining in popularity: Dublin Core. *Dublin* in this case refers not to the capital of Ireland but to the city in Ohio where the Ohio College Library Center (OCLC — now known as the Online Computer Library Center) is. The Dublin Core metadata set was developed at an OCLC workshop. It was originally intended for the library community, but it has been incorporated into standards used by other types of institutions concerned with content, such as educational institutions and magazine publishers.

Dublin Core is quite general in its scope and therefore widely applicable, but it's not considered comprehensive enough for any one publication type in particular. A few other discovery metadata standards have arisen that are more specific to certain publication types:

♦ **Magazines:** PRISM (Publishing Requirements for Industry Standard Metadata) centers on magazine publishers and their online operations. It uses the Dublin Core bibliographic metadata set and builds on top of it. Version 1.0 of PRISM was released in April 2001; see `www.prismstandard.org` for details.

♦ **Books:** ONIX (Online Information Exchange) is being widely adopted among book publishers and retailers such as Amazon and Barnes & Noble. ONIX describes physical books; it includes fields for such things as the cover image, number of pages, and physical size of the book. ONIX does not apply to online content, although eBook extensions to ONIX are currently in the works. ONIX was originally unveiled in January 2000; see `www.editeur.org/onix.html`.

♦ **Scientific journals:** CrossRef (`www.crossref.org`) is increasingly used in the scientific journal community for reference linking in online journals. It specifies a set of bibliographic metadata for journal articles that includes DOIs. Researchers can click CrossRef links in online journal articles to look up full bibliographic citations of other articles; then they can use DOIs to get to the articles themselves. The CrossRef system went live in June 2000.

♦ **Educational materials:** The Learning Objects Metadata (LOM) scheme, overseen by the IEEE (Institute of Electrical and Electronic Engineers) Learning Technology Standards Committee, covers educational materials, ranging from lecture notes to entire courses. LOM also builds on Dublin Core for bibliographic metadata; see `ltsc.ieee.org/wg12/index.html`.

♦ **Music:** Two vendors, Schwann (`www.schwann.com`) and MUZE (`www.muze.com`), control competing proprietary metadata schemes for information about recorded music, such as artists, song and album titles, and release dates.

♦ **News stories:** NewsML is an emerging standard for exchanging news content items — including stories, illustrations, photos, sound clips and video — that is expected to be adopted by wire services worldwide (such as the Associated Press and Agence France Presse). Developed by the International Press Telecommunications Council (IPTC), NewsML replaces the ANPA 1312 header format for wire stories that has been in use at newspapers and broadcast news stations for decades. Because of the high volume and widespread distribution of wire-service news feeds, NewsML is poised to become an important metadata standard for news items. It was released in October 2000; see www.xmlnews.org/NewsML/.

- **Multimedia:** MPEG-4 is a standard for multimedia content from the Moving Picture Experts Group, which has already given us important standards for audio and video compression. MPEG-4 is intended to include all sorts of metadata about multimedia objects. MPEG-4 allows rights information to be specified by means of generic metadata definition capabilities, and it allows DRM tools to be integrated with playback functionality by means of the Intellectual Property Management Protocol (IPMP) interface. Some DRM vendors have implemented their technologies so that they work with the MPEG-4 IPMP interface. See www.cselt.it/mpeg/standards/mpeg-4/mpeg-4.htm for details. Future MPEG standards, MPEG-7 (whose specification should be released by the end of 2001) and MPEG-21 (discussed in Chapter 3), also contain potentially important metadata standards for multimedia content.

Overall, it's hard to imagine a content description standard that is comprehensive enough to meet the needs of all content industries without being impractically complex. Nevertheless, one such project called *<indecs>,* for Interoperability of Data in E-commerce Systems, was released in March 2000. The <indecs> creator is Godfrey Rust of MUZE in the UK (see www.indecs.org/ for details).

Rights

Rights metadata schemes can be standardized by adopting a rights model, as discussed in Chapter 4. But as we mention in Chapter 4, real-world rights specifications are often hard to pin down in a formal rights model, and many uses of content (such as fair use) cannot be accurately represented by a rights model.

This implies that a truly comprehensive rights model would have to be large and powerful in order to be useful. The most comprehensive open standard rights model available now is XrML, which is discussed in detail later in this chapter in the section "Extensible Rights Markup Language." A useful subset rights model is embodied in the Information and Content Exchange (ICE) protocol, which deals with business-to-business content syndication. ICE is also covered in detail later in this chapter, in the section "Information and Content Exchange."

A lightweight rights model is also included in the PRISM standard for magazines. The standard includes a set of rights description fields designed to be read and understood by humans rather than machines — that is, they are really just text descriptions of the rights available on a given piece of content (for example, a magazine article).

The more recent MPEG standards for audio, video, and multimedia include hooks for rights models and rights management tools, although they do not specify rights information per se. MPEG-7 also includes rights information, but not in detail — its content management model includes pointers to detailed rights info, which are outside the scope of the standard (and possibly intended to be the domain of a language such as XrML).

Functions without standards

As implied earlier, a few additional aspects of DRM do not invite standardization because they are unique to each publisher or a source of competitive advantage. These areas are

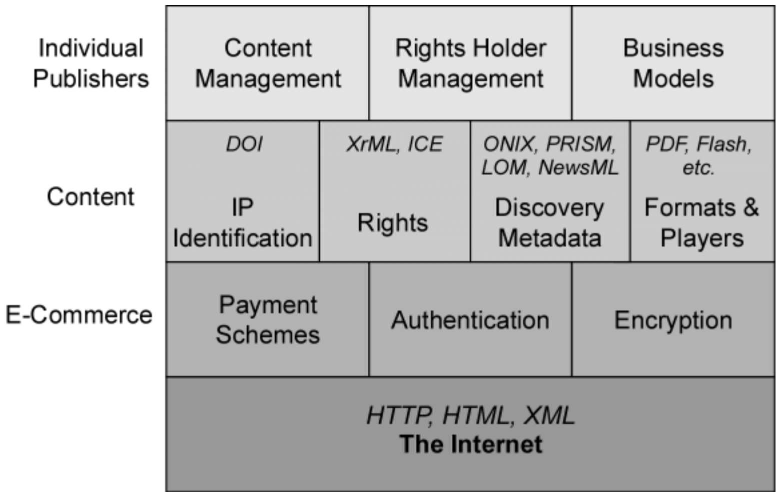
- **Content management:** As covered in Chapter 10, it's vitally important for publishers to deploy repositories of content in digital form that are accessible to DRM systems through APIs or other interfaces, enabling the DRM systems to get the content that they need automatically. Content management schemas and system implementations are unique to each publisher because they reflect the publisher's brands, organizational structure, and internal operations.

- **Rights and rightsholder management:** As implied in Chapter 4, relationships between publishers and other rightsholders — authors, agencies, and other publishers — are quite complex and hard to model. Furthermore, each publisher has a different way of processing rights and rightsholder information and transactions. Attempts to standardize on schemes for rightsholder management among publishers have largely failed (see Bill Rosenblatt's presentation, "Two Sides of the Coin: Publishers' Requirements for Digital Intellectual Property Management," in the bibliography), simply because the task is too complex — even more complex than content management.
- **Business models:** Of course, each publisher derives competitive advantage from the business models that it creates and implements. Publishers will never want to standardize on business models.

The DRM standards hierarchy

Figure 6-1 summarizes the preceding discussion by showing various layers of DRM-related standardization and how they relate to the basic standards of the Internet. The lowest layer shows the bedrock Internet standards, HTTP, HTML, and XML. The XML metalanguage forms the basis for many of the DRM-related standards at higher levels, including XrML, ICE, PRISM, and NewsML. The next layer up shows components of DRM that are necessary but not unique to DRM or to the media/publishing industries. These are mostly e-commerce elements, such as transaction processing, encryption, and authentication.

The third layer (second from the top) is where most of the action is. These are standards that are specific to the content industries, and they are as we just specified. Some are pure metadata standards, whereas others are protocol standards that contain metadata models. The principal ones are DOI for content identification, XrML and ICE for rights metadata, industry-specific discovery metadata standards, and the various popular content formats. The top layer of the diagram shows those elements that are unlikely to ever become standardized.



Copyright 2001 GiantSteps Media Technology Strategies

Figure 6-1: The hierarchy of DRM standards.

In the remainder of this chapter, we look at the most important of these standards initiatives in detail.

The Digital Object Identifier

The Digital Object Identifier (DOI) began in 1994 as part of a general online copyright management initiative within the Association of American Publishers (AAP) — the U.S. trade association for book and journal publishers. The initiative began as an attempt to find standard ways of solving the problem of online copyright management. At that time, the problem was focused on copyright protection on the Internet, as opposed to the broader problem of rights management that this book addresses. Nevertheless, DOI applies just as well to the latter as to the former.

After commissioning a study by the consultant Christopher Burns on publishers' attitudes towards online copyright issues (see Burns in the bibliography), the AAP's Enabling Technologies Committee broke the online copyright management problem down into layers similar to those discussed in the section "Types of standards relevant to DRM" earlier in this chapter.

The AAP's Enabling Technologies Committee decided to concentrate initially on the problem of content identification because the rest of the potential areas of standardization were too complex or varied too much from one type of publisher to another (for example, elementary through high school education versus scientific journals). The committee listed a number of requirements for online content identifiers. These are adapted from Bill Rosenblatt's December 1997 paper, "The Digital Object Identifier: Solving the Dilemma of Copyright Protection Online" (`www.press.umich.edu/jep/03-02/doi.html`), which gives more detail about the early history of the DOI project:

- ♦ The identifiers must be as "dumb" as possible; that is, they will have no intrinsic meaning.
- ♦ Identifiers must uniquely identify content items; there can be no duplication.
- ♦ There can be an infinite number of identifiers, with no limitation. Publishers will want to assign identifiers to individualized, potentially ephemeral products created on-the-fly.
- ♦ The identifying scheme must be a *metascheme* that will subsume any existing scheme that publishers may be using already, such as ISBN, ISSN, and so on, as well as Web URLs.
- ♦ Unlike the schemes mentioned in the section "Identification," earlier in this chapter, the identifying scheme can be used for content of any type, including static print-like material, services (for example, subscriptions or time-based access to content), software, audio, video, and so on.
- ♦ The scheme can also apply to objects of any granularity, ranging from the online equivalents of multivolume sets down to individual paragraphs or illustrations. Granularity decisions will be left entirely up to the publisher. Furthermore, an identifier can apply to a large object composed of smaller objects, each of which has its own identifier.
- ♦ An identifier must persistently refer to a content item regardless of its physical location or ownership, even if either are transferred to another publisher.

The committee created a specification for an identification scheme that met all these characteristics — the Digital Object Identifier.

DOI technology

The easiest way to understand DOIs technologically is to think of them as URLs that are permanent and location independent. The problem with using URLs as identifiers of intellectual property is that they point to locations of specific files on specific computers, which can change over time. If a publisher reorganizes its Web servers or (as happens regularly) sells a line of content to another publisher, the URLs are no longer valid, and anyone trying to browse to them gets the dreaded HTTP 404 (`File not found`) error.

DOIs, in contrast, point to entries in a huge table called a *DOI directory.* The table entry for a DOI specifies a URL to which the user will be referred; the publisher can change the URL at any time. So if a publisher changes its Web server architecture or sells an imprint to another publisher, it simply needs to change the URLs in the DOI directory entries for its DOIs.

The syntax of a DOI is shown in Figure 6-2, which shows an actual DOI for an article in the February 2000 issue of the journal *Growth Hormone and IGR Research,* published by Churchill Livingstone, a medical imprint of the publishing company Harcourt International.

A DOI has two components: a prefix and a suffix, separated by a forward slash (`/`). The prefix starts with `10`, which denotes the number of the DOI directory used to look up this DOI. (Currently there is only one directory, but this allows for future expansion.) The second part of the prefix is a number that identifies the publisher who created and registered the DOI. A publisher can have one or more prefixes, perhaps one for each brand or product line. In the case of Figure 6-2, the second part of the prefix is `1054`, denoting Churchill Livingstone; other Harcourt imprints have other DOI prefixes. Note that if the content changed hands — say, if Harcourt sold Churchill Livingstone to John Wiley & Sons — the prefix would stay the same; it would not change to `1002`, which is Wiley's prefix. The prefix stays the same to guarantee uniqueness of the overall DOI; if the prefix were to change, the overall DOI could change to something that already exists.

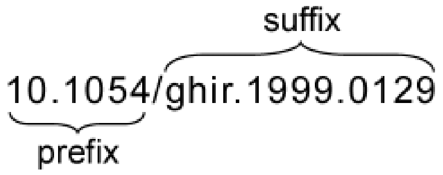


Figure 6-2: An example of a Digital Object Identifier. The prefix identifies the publisher that registered it, whereas the suffix is a character string of the publisher's choosing.

The suffix is a character string of the publisher's choosing; it has no special meaning to the software that processes DOIs. The string can contain letters, numbers, and most punctuation characters. The only requirement is that it be unique. The combination of a unique publisher prefix and suffix ensures overall uniqueness of DOIs across all publishers. In Figure 6-2, the suffix is `ghir.1999.0129`, which is mnemonic for *Growth Hormone and IGR Research* (the name of the journal), 1999 (year), and 0129, a number that is meaningful only to the publisher.

Many publishers have converted their existing numbering schemes, such as ISBNs and ISSNs, into DOIs simply by prepending their publishers' prefixes. DOIs can also be used at any granularity of content that the publisher wants, from complete works down to individual paragraphs or sentences if desired. (Pragmatically, most publishers are assigning DOIs at the levels of book, chapter, or journal article.)

A publisher starts the DOI process by paying a fee and obtaining a prefix for itself through a DOI *registration agency.* (Registration agencies are discussed later in this section.) After the International DOI Foundation (IDF, the organization that governs DOI — see `www.doi.org/`) grants the prefix, the publisher can start registering DOIs, which it can also do through a registration agency. For each DOI that the publisher wants to register, it submits a suffix to the registration agency along with the URL to which the publisher wants the DOI to point. The registration agency installs the DOI along with its corresponding URL in a central DOI directory and charges the publisher a fee for doing so. The publisher can subsequently change the URL associated with the DOI. (DOIs can't be deleted; they are considered to be permanent.)

It's important to remember that the URL need not point to actual content: It can also point to some software code that does various things, such as the following:

- Requires the user to log in, pay, or register before she can see the content
- Downloads a secure container that has the content in encrypted form
- Executes a database query, the result of which is the content requested
- Offers the user a subscription to the content
- Informs the user that the content is no longer available

After the publisher has registered DOIs, it can use them in a number of ways. Figure 6-3 shows one of them, in which the DOI directory server `dx.doi.org` is used to look up a DOI and then redirect the user's browser to the URL to which the DOI points. This process is called DOI *resolution.*

In step 1, the user is using a Web browser and clicks a link to the (hypothetical) URL `http://dx.doi.org/10.1346/ejournal-09-97`, denoting the September 1997 issue of *EJournal.* This syntax leads to a server, `dx.doi.org`, which treats everything after the `.org/` as a DOI and does a directory lookup.

In step 2, the DOI directory lookup finds the DOI `10.1346/ejournal-09-97` and retrieves the corresponding URL, `http://www.giantsteps.com/ejournal/purchreq?09-97`. Step 3 shows the DOI directory going to the `www.giantsteps.com` Web server and retrieving the URL, which turns out to be not the actual journal article but a CGI program that presents the user with an offer to purchase the content (step 4).

The technology underlying the DOI directory and DOI resolution is an existing system called the handle system, which was developed by the Corporation for National Research Initiatives (CNRI), a nonprofit research organization dedicated to inventing new technologies that enhance the Internet. CNRI is led by Dr. Robert Kahn, one of the two inventors of the TCP/IP protocol that forms the basis for the Internet. During the process of defining the DOI and determining how the technology was going to be implemented, the AAP committee members found that CNRI's handle system provided a very close match — a superset, in fact — to the desired functionality.

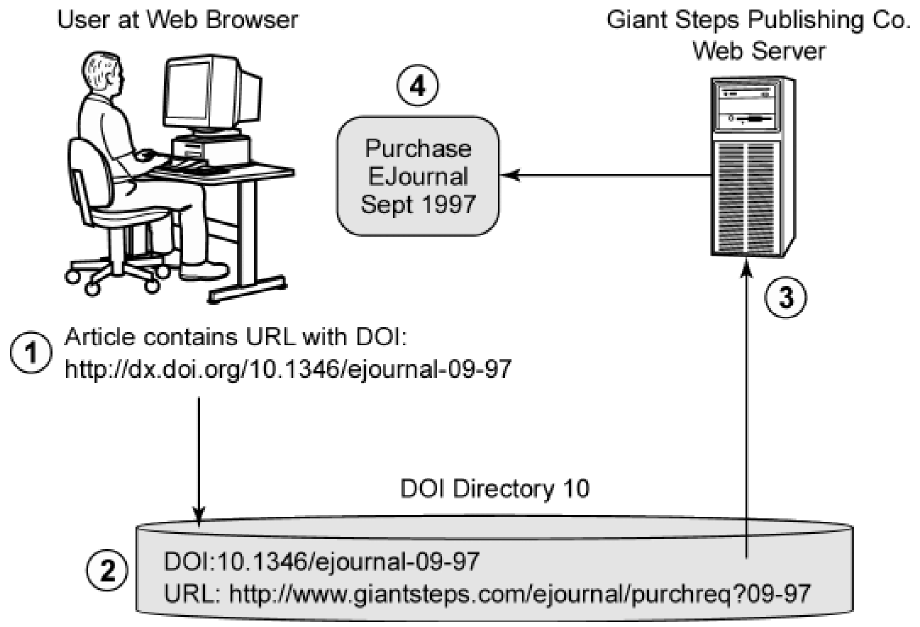

User at Web Browser
Giant Steps Publishing Co.
Web Server
4
Purchase
EJournal
Sept 1997
3
1
Article contains URL with DOI:
http://dx.doi.org/10.1346/ejournal-09-97
DOI Directory 10
2
DOI:10.1346/ejournal-09-97
URL: http://www.giantsteps.com/ejournal/purchreq?09-97

Figure 6-3: An example of the use of a DOI. The DOI Directory looks up the DOI and returns a URL.

CNRI operates the DOI directory and initially served as the only DOI registration agency available to publishers who wanted to register DOIs. At this writing, there are three others:

- ♦ CrossRef (see the section "Discovery metadata," earlier in this chapter), a nonprofit organization that uses DOIs to promote the linking of article references in scientific journals (`www.crossref.org/`).
- ♦ Content Directions, Inc. (`www.contentdirections.com/`), a New York–based startup company that offers DOI-related consulting as well as registration agency services.
- ♦ Enpia Systems Co., Ltd. (`www.enpia.com/`), an Asian digital content distributor based in Korea.

A company that wants to become a DOI registration agency must obtain certification from IDF. Publishers can set up their own registration agencies if they want to.

The example in Figure 6-3 shows how a DOI can be embedded in a standard URL on a Web page, but there are other ways to use DOIs. At worst, someone who knows a particular DOI can go to the site `http://dx.doi.org/` and type or cut and paste the DOI in — just as someone can type a URL in to a Web browser's Address bar. Beyond that, CNRI's handle system includes a *handle resolver,* which is a free piece of software available for download at `www.handle.net/resolver/index.html`. The handle resolver is a browser plug-in for Netscape and Microsoft browsers that implements the Web browser protocol `doi:` so that the browser can process URLs of the form `doi:`*`prefix`*`/`*`suffix`* (for example, `doi:10.1346/ejournal-09-97`) as opposed to `http://dx.doi.org/10.1346/ejournal-09-97`. The former is more efficient. CNRI has also created a handle resolver for wireless Internet connections through Palm handheld devices such as the Palm VII.

Another likely use of DOIs is embedding them into DRM software. Every DRM solution has some sort of field denoting an identification code for the content. Various vendors of DRM container and rights clearinghouse technologies have made it clear that it would be easy to use DOIs for this field. Therefore, in complete DRM solutions, a user could download some metadata that includes a DOI, which could be used to communicate with a secure server to get the content. Various vendors are reputedly working on DRM systems that include DOI registration agency functionality, but none have come out on the market yet. (One of them, Digital Goods, ceased operations in May 2001.)

Another possibility that has been discussed, but not yet implemented, is to build DOIs into metadata services that help users locate content. Users could search libraries in specific subject areas and get search results that include, say, abstracts of articles. DOIs would then link to the actual content. The CrossRef initiative is a simpler version of this: In the online journals of publishers participating in CrossRef, DOIs are used along with reference citations. These make it much easier for researchers than simply citing the usual bibliographic information, such as volume, issue, and page numbers.

DOI status

As mentioned previously, DOI is under the jurisdiction of the International DOI Foundation, a nonprofit organization founded in 1998. IDF has about 50 members, spanning educational and professional publishers, technology vendors, trade associations, and scholarly organizations. The DOI syntax has been accepted as a standard by ANSI and NISO (ANSI/NISO Z39.84-2000).

IDF has been working closely with CNRI and others to further the development of the DOI system along several fronts. CNRI’s original handle system, on which DOI is based, has the capability of resolving handles (its name for DOIs) to multiple URLs, not just single URLs. IDF is currently expanding the DOI system’s functionality to take advantage of this feature.

The vast majority of the early adopters of DOI have been scientific and educational publishers. DOI can certainly apply to all kinds of other media — indeed, to all other content on the Internet. IDF and other organizations have been evangelizing DOI to audio and video producers, financial research publishers, and so on, but the interest in those communities remains to be seen. IDF is also looking into ways of extending the reach of DOI into metadata schemes.

The eBook industry is looking seriously at DOIs in various ways. The AAP commissioned a study in 2000 that recommended the use of DOIs as the primary means of associating metadata with eBook content, and an initiative called DOI-EB (chaired by Steve Mooney) has been launched to foster the development of DOI applications for eBooks.

DOI is a simple yet powerful idea, and it is vitally important in the process of helping the DRM industry grow through open standards. There is a real need for what IDF calls an “actionable identifier,” that is, an identifier for content that is not only unique but also allows content to be found, subscribed to, purchased, and so on online and that can identify any type of content. It was clear to the AAP committee that created DOI that no existing type of identifier would satisfy all those requirements.

The sheer numbers of organizations adopting DOIs and becoming members of IDF suggest that DOI has a bright future — at least in publishing, if not across all content industries. And with companies such as Microsoft, Hewlett-Packard, and several DRM vendors as IDF members, DOIs are bound to make their way into the next generation of DRM-related technology solutions.

Extensible Rights Markup Language

Extensible Rights Markup Language (XrML) is a language for building rights specifications, as defined in Chapter 4. XrML has its origins in the work that Dr. Mark Stefik did at Xerox PARC research labs in the mid-1990s. As discussed in the Introduction, Stefik's work focused on his concept of trusted systems. *Trusted systems* are conceptual "black boxes" that can render content but only according to a precise definition of the conditions under which it should be rendered. Stefik decided that a trusted system needed a formal, standardized way to specify those conditions, which include the rights model issues discussed in Chapter 4. So he invented a language for doing so — the Digital Property Rights Language, or DPRL.

Stefik's idea is that any type of trusted system can be built so that it can read specifications written in a certain language and act on them accordingly. He originally imagined trusted systems to be hardware devices, but in reality, trusted systems can be DRM software on PCs, as well as hardware devices such as eBooks, digital music players, PDAs, smart cards, and so on.

Another important concept in Stefik's research is that of a *digital repository,* which is a collection of *documents,* which in turn include both content (presumably in encrypted form) and rights specifications. A digital repository can be a file system on a computer, the memory in a PDA, or a digital memory card such as those used in digital cameras and some music players.

As mentioned in Chapter 4, Stefik's background included research in object-oriented languages; he had designed an object-oriented version of the LISP programming language. Therefore, it's no surprise that DPRL originally resembled LISP in its syntax.

Xerox Corp. patented DPRL. As Xerox has done with many technologies from its renowned research lab, Xerox attempted to commercialize DPRL, forming a business unit called Xerox Rights Management to take it to market. Xerox Rights Management developed commercial DRM technology around DPRL and attempted to sell it to publishers with mixed success. It also created a second version of DPRL with an XML syntax instead of one derived from LISP.

In early 2000, Xerox spun Xerox Rights Management off into a separate company called ContentGuard, Inc. Xerox took a large stake in the new company; Microsoft also became a significant minority investor. ContentGuard took over all the intellectual property of Xerox Rights Management, including the patents on DPRL — which it modified and renamed XrML to reflect its XML foundation.

In sharp contrast with DOI, which is simple conceptually and syntactically, XrML is a large, rich language, with complexity roughly equivalent to the SQL programming language for databases or perhaps the PostScript language for print page description. But XrML is not quite a programming language; it is a *specification* language. That is, its intent is to allow programmers to specify the form or structure of something in detail without having to specify how that form or structure is actually implemented. This is true to the principles of XML, which is supposed to allow the structure of content to be specified without saying anything about how that content is rendered on a specific output device.

XrML enables programmers to specify how trusted systems communicate with other trusted systems, or with the "outside world" (for example, output devices), in terms of the exercise of rights. The language is meant to do so in enough detail that media companies, software vendors, and device manufacturers can create content, software, or hardware that implements the specifications accurately and unambiguously — just as it is up to printer makers to implement PostScript drivers or up to database vendors to implement SQL interpreters.

Those who want to implement XrML need to develop an *interpreter* for the language, which is a piece of software that takes commands in XrML and makes them run on the device in question, such as a PC, music player, PDA, eBook, or whatever. An XrML-enabled device can accept files (documents) written in the language and understand how to handle them, just as a PostScript printing device can interpret PostScript files.

For example, XrML has ways of defining rights on content, such as `PLAY` and `COPY`. An XrML interpreter must translate the `PLAY` right into code that actually plays the content on the device when the right is invoked. That's a straightforward example, but in order to make it work, XrML includes the definition of protocols between trusted systems that involve such "ugly" things as public-key cryptography, levels of security, and the details of file systems. Implementations of XrML must address these ugly aspects because without them, trusted systems would not have the security that they need.

All this is a way of saying that XrML has a lot of grungy details that won't be of interest to the reader who wants to understand what the language does and where it applies, without wanting or having to program in it. XrML code, though readable to anyone familiar with both XML and the ideas in this book, is intended to be read by machines, not humans.

Therefore, this section covers most of XrML's capabilities without getting into those ugly details. It would take an entire textbook to do justice to the language and provide a wide range of sample code. Anyone who wants to dig deeper can look at the XrML specification, version 1.03 at this writing, which is available at `www.xrml.org/` and which has plenty of examples in it. We provide a high-level overview of the language's capabilities here and refer the reader to the specification for further information.

XrML technology

A specification written in XrML is called a *document.* Don't confuse this with the actual content; it's really a bunch of metadata. The most important part of an XrML document is the `WORK` component.

The WORK component

The `WORK` component includes the content item's rights information and other metadata. It contains the following subcomponents:

- `OBJECT`
- `DESCRIPTION`
- `CREATOR`
- `OWNER`
- `DIGEST`
- `PARTS`
- `CONTENTS`
- `COPIES`
- `COMMENT`

- SKU
- RIGHTSGROUP (or REFERENCEDRIGHTSGROUP)

Of these, RIGHTSGROUP is the most interesting.

CROSS REFERENCE: We cover this topic shortly in the subsection, "Rights specifications."

OBJECT is the name of the work (for example, *Digital Rights Management: Business and Technology*) and an ID, which can be a DOI, ISBN, or other type of identifier. DESCRIPTION is an optional description of the work, mainly meant for human readability. CREATOR is a description of the work's creator, who can be an author, composer, or photographer. OWNER allows the creator of the XrML document to optionally specify a principal who is authorized to make changes to the rights of the content.

DIGEST is a cryptographic value, similar to a digital signature, that allows the creator to verify the integrity of the actual content — as opposed to SIGNATURE, which helps verify the integrity of the overall XrML document.

PARTS allows the XrML document creator to specify that other works are included as parts of the work that this XrML document describes. Conversely, CONTENTS describes the part of the actual content to which the rights specification applies. The way CONTENTS is described depends on the type of content it is; it can be expressed in bytes, SMPTE (Society of Motion Picture and Television Engineers) time codes, and so on.

COPIES describes the number of copies of the content to which the rights apply; if COPIES is omitted, the number is assumed to be one. This can be important because it influences transport rights (see the subsection "Types of rights"). If there are two copies of the content and the user exercises a LOAN right to lend a copy to another user, a second copy remains for the first user to use while the first one is lent out.

COMMENT is a text field reserved for comments, typically written by the person or application that created (or edited) the XrML document. As with comments in other programming languages, the comments aren't intended to be meaningful to any system that reads the document — just to the humans that read it.

Finally, SKU (a term borrowed from the retail world) is useful for retailers or distributors of digital content. It helps them relate the content to other things needed to sell it, such as cryptographic keys or discount coupons.

Rights specifications

RIGHTSGROUP specifies rights to the work. There can be several of them per document. RIGHTSGROUPS can have names, for example, "standard," "subscriber," "student," and so on. The main component of a RIGHTSGROUP is a RIGHTSLIST.

The RIGHTSLIST embodies the rights modeling power of XrML. The following rights are supported:

- **Render rights:** PLAY, PRINT, EXPORT, and VIEW
- **Transport rights**: COPY, TRANSFER, and LOAN
- **Derivative work rights:** EDIT, EXTRACT, and EMBED

The preceding rights map to the types of rights discussed in Chapter 4. In addition, XrML supports several types of rights that are called *utility rights* in Chapter 4:

- **File management rights:** `BACKUP`, `RESTORE`, `VERIFY`, `FOLDER`, `DIRECTORY`, and `DELETE`
- **Configuration rights:** `INSTALL` and `UNINSTALL`

Each right in a `RIGHTSLIST` has a set of terms and conditions associated with it, consisting of access controls, time periods, geographies, and considerations associated with exercising the right. In the following subsections, we examine the types of rights that XrML supports and then take a look at how terms and conditions are specified.

Types of rights

Mark Stefik's research identified fundamental differences between the types of render rights supported by XrML. `PLAY` means to create an *ephemeral* rendering of the content, one that goes away when the content is finished. (For example, the song or video is over.) `VIEW` is merely a synonym for `PLAY`, but it can apply to media to which the term *play* isn't particularly meaningful, such as still images. `PRINT` means to create a *permanent* rendering of the content on some output medium, such as hard copy. `EXPORT` means to pass the content out of the trusted system in plain, unencumbered digital form.

Transport rights in XrML refer to the same set of transport rights discussed in Chapter 4: `COPY`, meaning to create another copy of the XrML document; `TRANSFER`, meaning to move the XrML document from one digital repository to another; and `LOAN`, meaning to create a temporary copy of the document and move it to another digital repository for a specified period of time during which the original copy is inoperative (although, as explained earlier, if the XrML document specifies more than one copy, the other copies can still be operative, unless they too are loaned out).

Each of the transport rights has an optional clause called `NEXTRIGHTS`, which enables superdistribution. `NEXTRIGHTS` describes the rights to be added to or deleted from the new copy of the content; for example, the new copy may have its `COPY` right deleted, or its rendering fees can be marked up (a new `PLAY` right added). If `NEXTRIGHTS` isn't specified, the new copies of the content have the same rights as the original copy. The `LOAN` right also has a `REMAININGRIGHTS` clause, which is a convenience for specifying the (reduced) rights that the loaned copy of the work has.

Derivative work rights, as discussed in Chapter 4, are for using pieces of the work in another work. As mentioned before, it is not possible for any rights language to model all possible uses of content items. The designers of XrML admit this and suggest that the derivative work rights included in the language are meant only to cover the types of rights that are easiest to automate.

`EXTRACT` gives the user the right to take a portion of the work and create a new work, which in turn can be incorporated into a larger work. The new work that the `EXTRACT` right creates is represented in another XrML document. `EXTRACT` has an option, `EDITOR`, that lets you specify the type of editing software that you will allow to do the actual extraction. Ideally, `EDITOR` is another trusted system that knows how to create XrML documents. If `EDITOR` is omitted, as would be the case in most real-world applications, you can use any program to do the extraction; for example, you can use a Perl script to extract a certain section of a text document automatically or Microsoft Word to do it manually. However, the ideal world to which XrML applies is one in which every process that operates on content is a trusted system – which Perl and Microsoft Word are not.

The EDIT and EMBED rights are variations of EXTRACT. EDIT allows for extraction plus the option to change the content. EMBED allows extractions that explicitly insert the new work inside a new composite work. All three types of derivative work rights include an optional NEXTRIGHTS clause, as with transport rights, supporting superdistribution of portions of the original work.

Of the file management rights, FOLDER and DIRECTORY are ways of manipulating hierarchical folders of works, allowing XrML's notion of a repository to map onto hierarchical file systems, such as those found in Windows, Macintosh OS, UNIX, and other operating systems. The FOLDER right lets you specify rights that span an entire folder; these rights add to or override the rights that each document specifies. For example, you can use a FOLDER rights specification to disallow EXPORT rights on any of the documents in the folder, even if some of them specify EXPORT rights. If you are familiar with UNIX or other multiuser operating systems, you can think of FOLDER rights as a more sophisticated version of directory permissions.

DIRECTORY rights allow the user to list contents of a folder and their characteristics, similarly to the DIRECTORY command in DOS or the ls command in UNIX. DELETE rights are self-explanatory, but note that (as with any right) you can attach terms and conditions, such as fees charged.

BACKUP and RESTORE allow you to make backups of documents for the explicit purpose of restoring them if the original document has a problem. You can specify terms and conditions to the restore, such as to limit the number of restores possible or to charge a fee for each restore.

VERIFY rights allow users to run a program (a verifier) on the work that verifies its contents, according to some sort of security scheme. Such a program can verify a hash value or that the content is encrypted correctly.

Finally, the configuration rights INSTALL and UNINSTALL are necessary when dealing with secure repositories of works. They don't apply to platforms such as PCs and Macs, where no special permissions are necessary to install software.

Terms and conditions

Four types of terms and conditions can be attached to all types of rights in XrML:

- **Times:** At what times are the rights valid?
- **Consideration:** What sort of transaction takes place when the right is exercised?
- **Access controls:** Who can exercise the right, and what credentials do they need?
- **Territory:** Where, in geographic or digital space, is the right valid?

Time specifications in XrML are quite flexible. They allow for three kinds of time intervals: fixed, sliding, and metered. Fixed intervals are specified as FROM and UNTIL. Sliding intervals specify blocks of time from the time when the right is first exercised until a certain deadline — for example, any six-hour block until the end of December 2001. Metered intervals are like fixed intervals except that the time is cumulative, not in a single block — for example, a *total* of six hours through the end of December 2001. Time units in XrML time specifications are stated in units from years down to seconds.

XrML refers to consideration, as defined in Chapter 4, as "fees and incentives." You can specify these in two basic ways: as currency (any ISO-defined world currency, with U.S. dollars as the default) or as *tickets.* Tickets are a way to create a sort of private currency for publishers who want to support other forms of consideration. They are a kind of digital work in themselves, in that they have rights associated with them and they are meant to be stored in trusted system repositories. When you “play” a ticket, you use it according to some rules; for example, you render it invalid, or you decrement some value stored in it.

Tickets can be used for many purposes. The most obvious example is to use tickets as coupons that are exchangeable for “merchandise,” that is, rights to digital works. You can issue tickets to premium subscribers that they can use to access content; those who are not premium subscribers must pay cash. You can also use tickets as frequent flyer–style credits for users who spend a lot of time on your Web site or are otherwise loyal customers. Tickets can be shared among a number of publishers, similar to Flooz or other electronic incentive programs.

Tickets in XrML can have expiration dates, and they can be metered, with the same meaning as for time specifications. It is also possible to invoke transport rights on tickets, that is, to copy and transfer them to other users.

XrML handles currency transactions by including information about users’ financial accounts, which can be exported to transaction processing systems. This makes it possible to do several useful things, such as verifying credit card transactions on the fly and allowing users to be credited (given incentive) for viewing content instead of being charged fees. (For example, a publisher may offer credits for viewing advertisements.)

Several types of pricing are supported in XrML:

- `PERUSE`
- `METEREDFEE`
- `BESTPRICEUNDER`
- `CALLFORPRICE`
- `MARKUP`

Per-use pricing is straightforward: It’s charged each time the user exercises a right. Metered-fee pricing is used with time interval specifications that call for a time limit on exercising the right, such as sliding or metered intervals. With metered-fee pricing, you can specify the time increment (for example, minutes or seconds) as well as the time unit used for computing the time actually used. For example, you can charge a dollar a minute for use of some content, but actually charge to the nearest second.

A best-price-under scheme allows you to charge a price that isn’t determined exactly until the account is settled with a user. To use best-price-under pricing in an XrML document, you specify a maximum price that the user can expect to pay. This allows publishers to implement such things as rebates and situations where the exact price isn’t immediately known. For example, say that best-price-under pricing is used in a trusted system, such as a portable music player, that isn’t normally connected to the Internet and the user is purchasing content in a foreign currency. The exact purchase price won’t be known until the device connects to the Internet and does a currency conversion. Best-price-under schemes cover this eventuality while giving the user some idea of what she will be paying.

Call-for-price pricing is similar to best-price-under, except that the price must be settled *before* the right is granted, not later when the account is settled. A price ceiling isn’t necessary in call-for-price pricing.

Markup pricing is used in composite works in which each component of the work has its own pricing. You can use it to specify a percentage that is added to the aggregate price of the components. For example, consider a secondary publisher that produces a monthly digest of articles for a subscriber that is chosen according to her interest profile. The secondary publisher can impose a 10 percent markup for the convenience of having all those interesting articles delivered in one package. Using markup pricing in XrML, the secondary publisher can take just its 10 percent and leave the transactions for each of the articles to their original publishers to process.

You can specify minimum and maximum charges to each of these types of pricing. This makes it possible to charge tiny amounts for viewing small units of content while ensuring that the value of the transaction isn't less than the cost of processing it. It also allows pricing schemes such as "A dollar per play for the first ten plays; subsequent plays are free" (that is, setting a maximum price of $10). There is no direct way in XrML to support tiered pricing schemes such as "Copies 1 to 100 are 10 cents each, copies 101 to 1000 are 8 cents each, and copies 1001 and up are 5 cents each."

Access controls in XrML are used to put limits on the users that can exercise the specified rights. Without them, anyone could access content, as long as she had the proper consideration — for example, as long as she paid fees or possessed tickets. Some examples of cases in which a publisher would want to limit access to specified classes of users are

- ♦ Adult material — no one under age 18
- ♦ Material available to paid subscribers only
- ♦ Material intended for registered students of XYZ University (or employees of XYZ Corporation) only
- ♦ Classified material — secret clearance required
- ♦ Material playable only by devices made by XYZ manufacturer
- ♦ Material playable only on a player owned by the same person who owns the trusted system repository

The access control feature in XrML lets you specify electronic credentials (certificates, licenses, and so on) for both the source and the destination of the work. The source is the trusted repository where the work is currently stored; the destination is the trusted system to which it's going, whether that's a file system, an output device, or some other piece of software. (For example, the destination can be a media player that confirms the ID of its user via a password, a smart card, or a biometric device such as an iris scanner.)

XrML's access control scheme also lets you specify relative strengths of security, enabling you to stipulate that users or devices wanting access to a work have *at least* a certain level of security. Using the `SECURITYLEVEL` feature, you can specify different security criteria and require that devices wanting to access the work meet or exceed them. For example, you can use `SECURITYLEVEL` to define security attributes such as "password strength," "physical security," "government security clearance level," "virus checking enabled," and so on. Then you can assign level numbers to those attributes so that they can be compared.

Of course, getting the `SECURITYLEVEL` feature to work requires inventing standards for several types of security and levels within those types and implementing those standards across many different devices and pieces of software. A good example of such a scheme already in place is the security levels that the U.S. government assigns to computing facilities that handle classified data, which are designations such as C2 and B1. (Lower letters and numbers mean higher security.) Standardizing on security levels in the consumer sector is another matter — a monumental task, but XrML has the capability to handle it after it is done.

The final type of term and condition element in XrML rights specifications is the `TERRITORY`. This allows you to further limit the applicability of a right to a geographic range or a digital domain. You can specify physical (country, state, city, zip code, and street) or digital (IP addresses and other network domain identifiers) addresses with `TERRITORY` clauses.

Watermarks and tracking

Chapter 5 describes a watermark as a marking that can be inserted into a piece of content in such a way as to be both impervious to removal and minimally intrusive to the content's rendering. XrML provides support for watermarking — not by actually doing the watermarking, but by being able to provide information to a watermarking function that it may need.

Watermarks typically provide ways of identifying content for the purpose of tracking its use. Many types of information can be useful in such an identifier, such as the name of the content, the name of the publisher, the time and date of publication or of rendering (playing or printing), the name of the user for whom it is being rendered, and so on.

XrML's `WATERMARK` feature allows two types of information to be passed to an actual watermarking program:

- Data that is known at the time of publishing, which can be a text string (for example, the title of the work and name of the publisher) or binary data (for example, some audio material to be embedded into an audio clip)
- Data describing the actual rendering of the content, which can include
  - The list of rights defined on the work
  - Information about the user who is rendering the content
  - Information about the device doing the rendering
  - Information about the institution that owns the rendering device
  - The time of the rendering
  - The number of copies being rendered

You can include `WATERMARK` specifications in rights specifications in XrML. If a right has a watermark specification, the specified information is passed to a watermarking program when the right is exercised. Normally, this applies only to render rights, but it's easy to see how watermarking can also be applied to some transport and derivative work rights as well.

Related to watermarking is the capability of tracking each use of content. XrML has a `TRACK` feature, which lets you specify an entity that does the tracking, such as a logging program, and what information to feed it. You can specify `TRACK` with any right so that the exercise of that right can be tracked.

XrML status

As implied earlier in this section, XrML is a complex and impressive piece of technology. The result of meticulous research, XrML is the only comprehensive rights language available. It is not too much to say that the field of DRM came into existence with the publication of Mark

Stefik's paper on the original DPRL language. There are many advantages to standardizing on a rights specification language such as XrML throughout the media industry.

However, it would also be fair to say that ContentGuard has some work to do to achieve such standardization. Although over 20 companies — including DRM vendors, publishers, clearinghouses, and powerhouse vendors such as Adobe and Hewlett-Packard — have endorsed XrML, at this writing only one vendor other than ContentGuard has actually been implementing it . . . although it's not just any vendor; it's Microsoft.

The first issue that ContentGuard must address is that of the openness of XrML. ContentGuard controls XrML through several patents that it holds on the technology. That makes it fundamentally different from standards such as PRISM, ONIX, ICE, or even XML itself, which are not under patent protection. Some DRM technology vendors are not comfortable using a technology that's ultimately under the control of a company that may be a competitor.

ContentGuard has positioned XrML as analogous to the Java programming language from Sun Microsystems. As Sun did with Java, ContentGuard wants to appear as a thought leader for the industry, making a key enabling technology available to all, while offering products based on the technology that may be direct competitors to products from other vendors who license it. This was a good position to take a few years ago, when Java's openness was viewed in a positive light. But more recently, there has been something of a backlash against Java, as Sun has fought to control its definition and evolution against competitors such as Microsoft and HP. The general consensus now is that regardless of how "standard" Java is — what standards bodies have it registered, what authoring process it has, or whether the source code is under an open-source license — Sun remains in control of the language, to the annoyance of certain other vendors.

The good news is that ContentGuard is licensing XrML freely. It is possible for anyone to get the XrML specification simply by registering for it on the `www.xrml.org` Web site, and anyone can implement the language without paying royalties. However, ContentGuard has not established an open authoring group for the evolution of the language (as all the other standards mentioned earlier have done), although it intends to do so.

The complexity of the language is another issue. XrML has lots of features, some of which overlap — meaning that there may be several ways to implement a certain business model in XrML. This isn't meant to be a criticism, though: The same is true of virtually all programming languages. Mark Stefik and his team at PARC talked to a wide variety of publishers and device makers in order to gather requirements for the original DPRL language, and they wanted to ensure that all the requirements were met. ContentGuard has done more of the same.

Yet XrML is so full-featured that an interpreter for the language has to be a large piece of software, one that takes up lots of expensive memory (for example, in a portable music player) or software download time. Furthermore, the language embraces several software elements with deep implications for the inner workings of the systems that implement it, such as public-key encryption and multidimensional security levels, which may conflict with features already built into (or intentionally omitted from) those systems. This also adds to the complexity of implementing the language, although such features are necessary to enforce rights. XrML was originally intended to support the entire trusted systems concept, not some lightweight subset thereof, and it's unclear whether it can be used successfully for the latter.

These are all problems that ContentGuard can fix, and indeed intends to fix. ContentGuard can evangelize XrML to the industry by creating an open authoring group, offering training and implementation assistance to vendors, holding developers' conferences, touting its benefits at other conferences, and so on. Remember that despite the minor backlash against Java, Sun evangelized it to the entire computer industry, and it's an undeniable success.

With so many DRM vendors having introduced incompatible, proprietary rights models, the need for XrML to succeed is great — but the industry has to be properly motivated for it to happen. We hope that ContentGuard is up to the challenge as it also builds the revenue-generating portion of its business.

Information and Content Exchange

The Information and Content Exchange (ICE) is a protocol for supporting content syndication relationships. It began as an effort within Vignette Corp., the makers of the popular StoryServer Web publishing system, to create technology that enabled Vignette's customers to exchange content. Instead of building a proprietary product within the company, Vignette decided to create an open standard in collaboration with some customers.

One of those customers was News Internet Services, the Internet arm of Rupert Murdoch's News Corp., whose vice president of engineering, Laird Popkin, ended up coauthoring the original ICE specification along with Vignette's Brad Husick. The first version of the specification appeared in October 1998.

ICE is concerned with the business-to-business (B-to-B) side of digital rights management. Throughout this book, we point out that although the media has paid the most attention to the business-to-consumer (B-to-C) side of DRM (covering the controlled distribution of content from publishers to consumers), B-to-B side is just as important, if not more so. The B-to-B side of DRM covers cases of publishers wanting to get content from each other. In Chapters 2 and 4, we include examples of book publishers using each other's content, particularly in the textbook area where publishers routinely license illustrations, tables, equations, and other small pieces of content from one another.

Another important example of B-to-B content transactions is among Web sites. A content Web site devoted to a particular area of interest may want to get some of its content from other sources. For example, a travel Web site may want to get restaurant reviews from a food Web site to augment its information about vacation destinations. A reseller of computer software may want to get documentation updates from the vendor whose wares it resells and put them up on its site. One of the great things about the Web is that it's so easy to include content from Site A on Site B: Just copy it or hyperlink it.

Yet doing this kind of thing on a regular basis has its logistical problems. Site B needs information such as where to get Site A's content, what format the content is in, when it is updated, what content items Site A has from which Site B can choose, and of course, what rights it has to display and redistribute Site A's content. These are some of the main elements in what we call a *syndication* relationship.

Historically, the term *syndication* has been most often associated with radio and TV programs, news stories, and cartoons. Major newspapers such as the *New York Times* and *Los Angeles Times* have their own syndicates. If the *Times* (either one) has a news story that it thinks may be of national interest, it puts the story up on its own syndication network. Subscribers to that network pick the story up and run it in their own newspapers if they choose. Companies such as King Features syndicate cartoons (such as "Beetle Bailey" and "Blondie") and columns (such as those written by Dr. Joyce Brothers and Dan Rather). In the world of television, the big studios have divisions that syndicate their programming to TV stations; for example, Buena Vista is Disney's syndication division.

Major content brands have begun to syndicate their content to other sites on the Web. They have discovered that the best way to increase brand recognition is not to try to draw the most users to a single Web site but to get their branded content to as many other sites as possible that attract the right kind of user. For example, *Business Week* magazine maintains its own Web site, *Business Week* Online, but also syndicates its content to several different places, ranging from general Web portals such as Lycos to business destinations such as Staples.com, the online store of the office supply retailing giant. *Business Week* also syndicates its content to traditional, pre-Internet online services such as Dialog and Lexis-Nexis.

If a publisher wants to syndicate its content to a certain number of different sites, it has to set up the same number of different logistical schemes for getting the content to them. Typically, such schemes have been set up with FTP (the standard File Transfer Protocol on the Internet) and a collection of ad hoc file formats. Such schemes require fairly major programming efforts every time the publisher wants to add another syndication partner. Clearly this is not a scalable activity.

ICE was invented to create a standard way of handling the logistics of content syndication. It does *not* handle two kinds of things that are common to consumer-oriented DRM solutions: business terms and copy protection. Terms of business relationships, such as pricing and legal liabilities, are assumed to be negotiated by humans and represented in contracts — and the parties in the contracts are assumed to trust each other to carry out their parts of the relationships appropriately. It's taken for granted that the subscriber to the content will pay for it (if necessary) and will not distribute unauthorized copies of it.

Nevertheless, ICE has a lightweight way of describing content rights, and it can interoperate with DRM solutions that specify rights in detail and that enforce copy protection. The authors of ICE didn't expect most B-to-B content syndication relationships to need heavy-duty rights protection, so they didn't build it in.

ICE technology

The ICE standard is currently being developed under the umbrella of IDEAlliance, a group within the Graphics Communication Association (GCA) that promotes open standards and oversees the PRISM standard for magazines and their Web sites. The definitive document on ICE is the ICE specification, currently version 1.1, available on the `www.icestandard.org` Web site. Another very useful document on the same Web site is the ICE Implementation Cookbook, which provides step-by-step examples of ICE-compliant implementations.

ICE is a communications protocol. Like XrML, it is expressed as an XML vocabulary. But a protocol is a different type of animal than a specification language, which is what XrML is. ICE not only determines how certain types of software should work (which is what a software specification does), but it also determines how these types of software should communicate with each other.

In ICE, two types of software communicate with each other: *syndicators* and *subscribers.* Syndicators make collections (*subscriptions)* of content available to subscribers over time by sending them packages periodically with instructions to add new content items to their collections or remove content items from them.

Figure 6-4 shows a basic example of this. The syndicator is Zingo, a restaurant reviewer, and the subscriber is TravelPace, a travel site. TravelPace has the Zingo subscription called “NYC Restaurant Reviews.” TravelPace’s collection of reviews, the result of various packages sent by Zingo previously, includes the restaurants Bistro Dordogne, Hoss’s Steakhouse, Les Amateurs du Vin, Trattoria Il Duomo, and Adobe Café (presumably among many others).

In Figure 6-4, Zingo is sending TravelPace its May 2001 package. The package contains commands to add reviews for the new restaurants Thanos’s Taverna, Bistro Dordogne, and Wursthalle Berlin. The review of Bistro Dordogne replaces the one that TravelPace already has because the restaurant got a new chef and the *pommes sarladais* are much improved. The package also contains a command to remove the review of Trattoria Il Duomo: The restaurant closed after the *New York Times* restaurant reviewer came down with salmonella after eating there. As a result of the operation, TravelPace has a new collection of restaurant reviews, which it can display on its Web site.

Using ICE, syndicators can do the following:

- Put up catalogs containing offers of content packages that they are making available for subscription, along with specifications about how often the packages will be updated.
- Accept messages from subscribers who want to subscribe to particular offers in the catalog and who want to negotiate terms of the subscription.
- *Push* content packages to subscribers according to predetermined schedules.
- Send unsolicited messages occasionally to subscribers — for example, to tell them about new content offerings or to inform them about scheduled service interruptions.

Meanwhile, subscribers can do the following:

- Browse syndicators’ catalogs, choose offers to subscribe to, and negotiate subscription terms.
- *Pull* content packages from syndicators’ Web sites.

ICE supports both push and pull methods of delivering content. It also defines two types of subscribers: weak and full. A *full* subscriber is assumed to have a server up and running at all times, ready to receive pushed content and unsolicited messages, as well as to pull content from the syndicator. A *weak* subscriber is one that only gets (pulls) content on demand, possibly on an ad hoc basis.

The basic unit of communication in ICE is a message called a *payload,* which is an XML document that conforms to the ICE DTD (Document Type Definition). Payloads contain messages that syndicators and subscribers pass back and forth. The messages use lots of unique ID numbers to keep track of things such as the identity of the syndicator, the identity of the subscriber, the type of message, the identity of a content package being requested or sent, and so on. The messages also contain either *requests,* which are commands being sent from the syndicator to the subscriber or vice versa, or *responses* to those requests.

Requests in ICE fall into two broad categories: those that have to do with syndicators’ catalogs and subscriptions to the content and those that have to do with subscribers getting content to which they have subscribed.

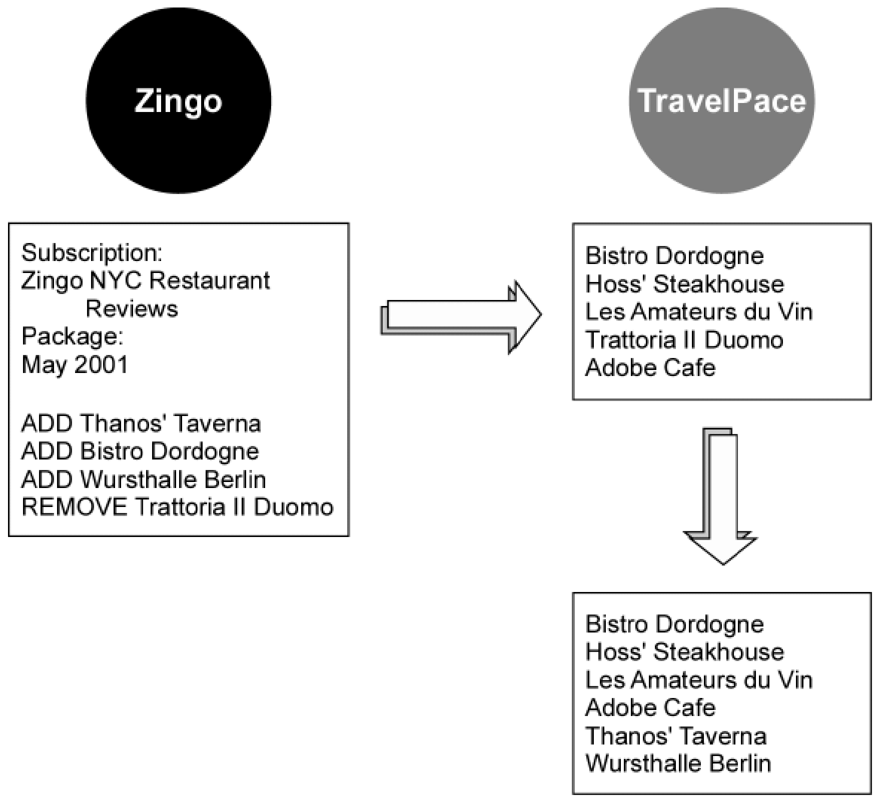


Figure 6-4: A syndication package in ICE.

Catalogs and subscriptions

The first thing that happens when a syndicator wants to make its content available is that the syndicator constructs a catalog of offers of content packages. A catalog consists of offers, which are descriptions of the content items available. Offers can be grouped into offer-groups, which can themselves contain offer-groups (allowing for hierarchical organization of offers).

Each offer contains several metadata fields, including an ID for the offer, a descriptive name of the product being offered, and fields that capture content rights information:

- Derivative work rights:
  - `atomic-use`: Tells whether the content must be used in its entirety or if subscribers can use parts. This is the converse of an *extract* right as defined in XrML and in Chapter 4.
  - `editable`: Tells whether the subscriber is allowed to edit the content before redistributing it. This is similar to an *edit* right as defined in XrML and in Chapter 4.
- `ip-status`: Specifies any licensing restrictions on the content. Possible `ip-status` values are
  - `SEE-LICENSE`: The content is covered under preexisting contractual terms; this is the default value if the field is omitted.
  - `PUBLIC-DOMAIN`: No licensing restrictions.
  - `FREE-WITH-ACK`: No licensing restrictions, except that the subscriber is required to display an acknowledgement string along with the content.

  - `SEVERE-RESTRICTIONS`: This is meant to allow ICE processors to flag content items for special attention if their licensing terms fall outside of the preexisting license agreement.
  - `CONFIDENTIAL`: The content is confidential and not meant for further distribution.
- `rights-holder`: Name of the entity that owns the syndication rights to the content.
- `show-credit`: If true, requires the subscriber to display the source of the content (as with the `ACK` above).
- `usage-required`: If true, requires the subscriber to return data on the usage of this content item.
- `quantity`: The number of times the subscriber is allowed to get an update of this content item; for example, if the content item is a stock quote on a given ticker symbol, the syndicator may limit the number of times a subscriber can access it.

An offer description also contains a *delivery rule,* which is a set of specifications about when the syndicator delivers packages to subscribers, and a list of *business terms,* which are pointers to explanations of different types of business relationships between the syndicator and subscriber.

Delivery rules contain several attributes that give syndicators a lot of flexibility as well as precision in defining delivery schedules for their content, including

- Whether the delivery is push (syndicator pushes content to subscribers) or pull (subscribers pull content from syndicator)
- The start and end time of the delivery
- The days of the week (Monday, Tuesday, Wednesday, and so on) or month (1st, 2nd, 3rd, and so on) that the delivery takes place
- The start time and duration of a specific time window on the above days when the delivery takes place
- The minimum and maximum time between updates of the content item
- The minimum and maximum number of updates of the content item

Business terms include the following:

- **Credit:** the required text crediting the source of the content that the subscriber must display
- **Licensing:** the terms of licensing
- **Payment:** the payment terms
- **Reporting:** the description of data on end-user content access that the subscriber is required to report
- **Usage:** the uses to which the subscriber may put the content

ICE doesn't contain any direct means of specifying the business terms. Instead, each of them can contain any of the following:

- A text string with the details of the business term, such as the credit string "Copyright 2000 XYZ Publishing, Inc."
- An ID number designating the business term, such as a reference to a payment schedule called `MONTHLYSUB`, as opposed to another one called `ANNUALSUB` or `SINGLEITEMSUB`.

- ♦ A URL that points to a file with the term's details; for example, the usage term can contain a URL pointing to an XrML description of the content's allowed usages.

This information framework for delivery schedules and business terms is intended to support automated negotiation between syndicators and subscribers. ICE contains an elaborate mechanism that allows the two parties to negotiate the details of these parameters.

For delivery schedules, a syndicator can specify which terms are negotiable and within what ranges. ICE includes a protocol by which subscribers can offer terms, syndicators can accept or reject them, and negotiation can proceed until either there is a set of mutually acceptable terms or there is no deal possible. ICE ensures this by requiring that only one parameter at a time is in play and that after a parameter value has been agreed to, it cannot be revisited later in the negotiation. A subscriber can offer a set of terms, and the syndicator can respond by saying, in effect, one of three things: "OK," "No, but feel free to try again," or "No deal is possible."

For business terms, the ICE specifications are merely a framework within which external software — or humans — can negotiate. There is no mechanism within ICE, for example, for negotiating the text of a credit line or the exact details of subscriber usage reporting.

ICE also allows the negotiation framework to be extended to include additional parameters, called *subscription extensions.* For example, a syndicator can specify a range of content formats from which subscribers can choose (such as Microsoft Word, PDF, or HTML for text or Real, Windows Media Player, or QuickTime for streaming video), or it can specify that an encryption-based DRM solution can be used to transmit the content.

When a subscriber wants to subscribe to content, it sends an `ice-get-catalog` message to the syndicator. The syndicator responds by sending the catalog in an `ice-catalog` message. Then both parties send a series of `ice-offer` messages until the negotiation is complete, at which time the subscriber has a subscription (assuming that the negotiation was successful).

After a subscriber has a subscription, it can cancel, ask for the susbscription status, or change the subscription. Changing the subscription starts a new negotiation process.

Content delivery

ICE provides a framework for delivering content that lets syndicators define ordered sequences of content packages and ensure that the packages are all delivered in the proper order. The framework is set up so that the syndicator has to keep track of the overall order of the sequence and where each subscriber is in the sequence; subscribers know only about the state they're in at any given moment. This gives syndicators a way of having more knowledge about their content than they may want to pass on to subscribers — which can be useful if, say, the content is a quiz sent as a series of questions, the number of which is to be kept secret.

Each package contains a lot of metadata, some of which overlaps with the information contained in subscriptions and overrides their values if they are set. Here are some of the metadata attributes unique to packages:

- ♦ `activation` and `expiration`: Start and end dates/times of the validity of the package.
- ♦ `fullupdate`: Specifies whether processing this package requires updating all the content that the subscription has delivered so far or (the default case) whether the package is incremental.
- ♦ `old-state` and `new-state`: Conveys state information by specifying the state of the package sequence immediately before (old-state) and after (new-state) this package is sent.

The subscriber is expected to know that it's in old-state before accepting the package; if it isn't, it can report an error.

Aside from the metadata, a package contains a set of ice-add and ice-remove instructions to add and remove content from the subscriber's collection. You can see the basics of how this works in Figure 6-4, earlier in this chapter.

The `ice-add` and `ice-remove` commands are associated with items. Items can be one of three things:

- ♦ Content (`ice-item`).
- ♦ A pointer to content (`ice-item-ref`).
- ♦ An item group (`ice-item-group`). Groups can contain groups, which makes it possible to organize items hierarchically.

`ice-items` contain the actual content within the ICE payload. They also include several metadata attributes that overlap with subscription metadata, such as rights information. Any attributes stored in `ice-items` override the corresponding attribute values of subscriptions. In addition, `ice-items` contain these attributes:

- ♦ `activation` and `expiration`: The start and end dates/times of the validity of the item.
- ♦ `content-filename`: This is a filename to be used as a destination name on the subscriber's site. It is often necessary to specify this attribute to ensure that file references in URLs across multiple content items are preserved properly on the subscriber's site.
- ♦ `content-type`: The MIME type of the content.
- ♦ `subscription-element`: An identifier for this content item that stays constant for the life of the subscription. This allows updates (adds where the content item already exists) and removes to work correctly.
- ♦ `update-attributes-only`: If true and this is an update (an add where the subscriber already has an item with the given `subscription-element`), don't update the content; just update the attributes.

`ice-item-refs` are more commonly used for content that doesn't comfortably fit into a delivered package, particularly streaming audio or video. They contain URLs that point to the content. An `ice-item-ref` can also contain a description of the time window during which the content is accessible, as well as access control parameters (such as user IDs, passwords, and cookie strings) that enable access to the content.

As mentioned previously, there are two ways of getting packages from syndicators to subscribers in ICE: push and pull. For push, syndicators send `ice-package` messages to subscribers, who are expected to be able to receive and possibly confirm the delivery of the packages. For pull, subscribers issue `ice-get-package` messages that contain the ID of the subscription.

ICE status

The ICE consortium has two levels of membership: regular member and authoring group. Each level has a handful of members, most of which are technology vendors. More importantly, several products on the market implement ICE. These are one flavor or another of syndication server — server products that enable content companies to syndicate their content and enable Web sites to subscribe to the syndicators' offerings. Some are designed as add-ons to existing

Web publishing systems or application servers, whereas others are standalone packages. Here are some current ICE-compliant products:

- ♦ Vignette Syndication Server: the original ICE server, designed to work with Vignette's StoryServer Web content management system
- ♦ HP Bluestone Total-e-Syndication: a Java-based application server component
- ♦ Kinecta Syndication Server: a standalone syndication server package
- ♦ Xenosys JICE: Java programming language components for ICE
- ♦ Intershop Enfinity: a Web content management system with ICE support
- ♦ Quark avenue.quark: a tool for exporting QuarkXPress page layouts to Vignette StoryServer in XML
- ♦ ArcadiaOne eSyndication: a standalone syndication server package
- ♦ Interwoven OpenSyndicate – a syndication server package designed to work with Interwoven's TeamSite Web content management system.

The products from Vignette, Kinecta, ArcadiaOne, and Interwoven are described in Chapter 12.

ICE's installed base of vendors and customers has been growing steadily. However, two forces act as barriers to ICE's further growth. The first is the expense and complexity of setting up ICE servers when compared to small-scale, ad hoc syndication. As mentioned previously, publishers who want to send their content to others have often been satisfied with simple FTP-based schemes. Some publishers who have been syndicating content for many years have a lot invested in legacy file formats for doing so: For example, McGraw-Hill has a standard tag format that they have used for many years to send content from titles such as *Business Week, Aviation Week,* and *Engineering News-Record* to online services such as Dialog and Lexis-Nexis.

Because publishers usually start out with one distribution partner and add others incrementally, it's unusual that one would think about building an architecture for scalable syndication from the start. More often, publishers build lots of ad hoc functionality and then, when it gets out of hand, decide to scrap it in favor of an extensible, robust architecture such as ICE. This kind of decision is neither lightly made nor easily executed.

The second barrier to ICE growth is the rise of third-party syndication networks such as Screaming Media and YellowBrix (see Chapter 12). These serve as hubs that sit between those who want to syndicate their content to others and those Web sites who want to use content from a variety of sources. The advantage of these services is that they make it very easy for publishers and subscribers — easier than adopting complex server software in-house. In particular, these services accept content in a variety of formats without requiring ICE-compliant payloads. (For example, Screaming Media accepts content in an XML tag set that is considerably simpler than ICE.) These syndication networks take the muss and fuss out of syndication. Of course, they do it for a price: They take large percentages of transaction revenue.

ICE remains the best solution for those publishers who want to build their own syndication architecture instead of trusting it to third parties. ICE enables them to build services that are more flexible and reliable than ad hoc FTP-type schemes, and it lets them keep the revenue in their own pockets. ICE is a vital part of the DRM standards landscape.

Secure Digital Music Initiative

The Secure Digital Music Initiative (SDMI) began in early 1999 as an initiative within the music industry that was analogous to the AAP’s Enabling Technologies project, which resulted in DOI. The music industry wanted to forge a set of open standards for the online distribution of digital music with built-in rights management, and at the same time, create a viable marketplace for music encoded in accordance with the standards.

The record industry’s trade association, the Recording Industry Association of America (RIAA), and representatives from the “Big 5” record labels that control the music industry (Sony, Warner, BMG, EMI, and Universal) called a conference of technology companies and consumer electronics makers in February 1999 to kick off SDMI. The initiative’s motivation was clear: The MP3 file format, enabling good-quality digital music to be freely distributed over the Internet, was threatening the foundations of the music industry. The record companies had to do something, and fast.

As mentioned at the beginning of this chapter, when a trade association tries to create standards, there are significant legal constraints on the kinds of standards that they can impose: They must be procompetitive, not anticompetitive. Thus, it was not possible for the RIAA and the record labels to simply designate some vendor’s technology as the endorsed standard for secure digital music distribution.

Instead, SDMI tried to set up a framework on which open, procompetitive standards could be built. This was the long-term goal. However, SDMI also needed a short-term goal to help stanch the flood of pirated music on the Internet. The SDMI participants decided not to concentrate on PC software, but instead to focus on portable music player devices, such as the cheap MP3 players that were beginning to flood the market at that time. Their short-term goal was to produce a specification for portable music players by June 1999 that consumer electronic makers could use, and they wanted the latter to get SDMI-compliant player devices out onto the market in time for that year’s Christmas season.

This goal was incredibly ambitious. To help achieve it, they hired as SDMI executive director Dr. Leonardo Chiariglione (pronounced “kyar-ee-LYOHN-eh”), a brilliant engineer from Telecom Italia who had chaired MPEG. A growing number of organizations, currently exceeding 200, joined the effort.

The first version of the SDMI portable device specification met its June 1999 deadline . . . almost. However, the 35-page document was not a specification that multiple manufacturers could implement with interoperability; it was too high-level. Think of a group tasked with inventing a new written and spoken language, with the objective of getting poets to write poetry in the language. They put out a description that defines nouns, verbs, adjectives, and rules for putting them together into sentences but says nothing about what alphabet to use or how to pronounce the words. Then they distribute this language description to poets and tell them to write poetry that everyone can read.

Thus the first version of the SDMI specification served as a statement of design principles rather than a blueprint for manufacturers, although it was sufficient for Phase I implementations (see the following section, “SDMI Technology”). A good summary of the design principles, as well as SDMI’s market objectives, is in the PowerPoint presentation “Secure Digital Music Initiative: Creating a Digital Music Marketplace,” available on SDMI’s Web site at `www.sdmi.org/download/create_dig_mktplace.ppt`. The specification itself, along with other supporting documentation, is at `www.sdmi.org/port_device_spec_overview.htm`.

Along with the release of the specification came announcements by many consumer electronics makers of the impending release of SDMI-compliant products, planned to be in time for Christmas shopping in 1999. Announcements came from the likes of Diamond (makers of the Rio MP3 players), Creative Labs, Matsushita (Panasonic), Toshiba, Mitsubishi, Lucent, Sanyo, Philips, Sony, Thomson (RCA), and Audiovox. SDMI created a logo and a licensing program for these vendors and any others who would sign on.

SDMI technology

The specification covers portable devices (PDs), which are the music players themselves; portable media (PM), which are the memory devices that store the music, and Licensed Compliant Modules (LCMs), which are hardware or software devices that process audio files and feed them to players and distribution channels.

LCMs are the main focus. Their job includes these functions:

- ♦ Convey audio from various types of media to PDs.
- ♦ Check for pirated content by testing for the presence of a watermark.
- ♦ Package content for secure distribution by using encryption and watermarking.
- ♦ Implement whatever business rules are defined — for example, for collecting payment or usage information from the consumer.

SDMI agreed that, initially at least, certified players would have to be allowed to play open formats such as MP3. This was necessary to avoid alienating consumers and device makers alike.

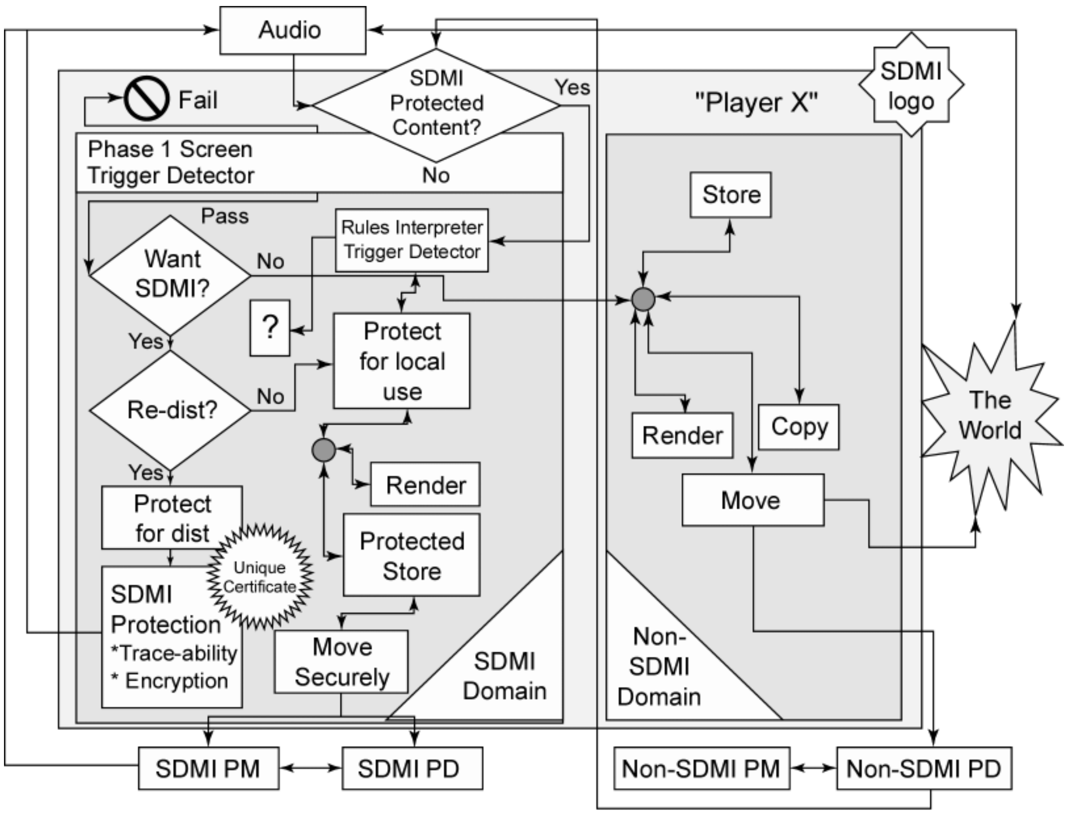
Figure 6-5, taken directly from the specification, shows how LCMs work. An LCM tests to see if the content is in an SDMI-compliant encrypted form. If it is, the LCM passes the content on to the business rules interpreter and eventually to an SDMI-compliant player or media device. If it is not encrypted, the LCM checks for the presence of a watermark, which in this case is an audio signal beyond the spectrum audible by humans. If the watermark is there, the content is legitimate; if not, it is rejected as pirated. Content produced by an SDMI-compliant device has a watermark that disappears if the content is copied by a non-SDMI-compliant device.

LCMs also have the capability of repackaging content for SDMI-compliant redistribution. To do so, they assign unique certificates, insert watermarks, and encrypt the content before making it available to the distribution channel, whether that's a media device, a file on someone's computer, or the Internet.

The SDMI architecture proceeded in two phases. In Phase I, players play music in any format; only in Phase II would players implement the protection mechanisms described earlier. Phase I players simply detect the SDMI watermark and, after Phase II devices become available, tell users to upgrade to a Phase II device. Eventually, Phase I devices would reject SDMI-watermarked content. Phase II devices would have all the watermarking and encryption technology necessary to implement the SDMI architecture.

In August 1999, SDMI selected watermarking technology from Aris Technologies for Phase I. (Aris has since merged with competitor Solana to form Verance Corp.) SDMI began Phase II by issuing a call for proposals from technology vendors in February 2000, from which it received over a dozen responses. SDMI chose five finalists, whose names have not been made public.

SDMI issued a public challenge (called “HackSDMI”) to break the security of the proposed solutions. This showed that they took important lessons from the cryptography field to heart: A cryptography-related standard is much more likely to catch on and endure if its algorithm is subject to public examination and testing instead of being kept secret.



The ? box represents the ability of an SDMI-compliant application to implement a variety of licensed operations, including requiring an upgrade to Phase 2.

Figure 6-5: The SDMI licensed compliant module architecture.

However, when a team of people from Princeton University succeeded in cracking some of the proposed technologies in the fall of 2000, the RIAA lost its resolve. Prof. Edward Felten, the lead researcher on the Princeton team, had written a paper on how he cracked the SDMI solutions, which he intended to present at the Information Hiding Workshop conference in Pittsburgh, in April 2001. But instead of encouraging the publication of Felten’s results as part of the natural process of strengthening SDMI’s technology, the RIAA tried to muzzle him by invoking the Digital Millennium Copyright Act (see Chapter 3). He agreed not to deliver the paper at that particular conference, but the ensuing firestorm of media attention cast the RIAA in an unfavorable light. As a result, the RIAA backed down and let Felten deliver his paper at the USENIX Security Symposium in San Francisco in August 2001.

SDMI status

At this writing, SDMI has lost much of its momentum. Deadline after deadline has been missed. Despite all the announcements by consumer electronics makers, it appears that Sony is the only vendor that has actually marketed an SDMI-compliant device, the Sony VAIO Music Clip. The device is Phase I compatible, meaning that it functions as a generic digital music player.

The Phase II screening technology selection process concluded in early 2001 without a selection having been made. Around the same time, Leonardo Chiariglione resigned as executive director. SDMI continues on, but at this writing, has yet to appoint another leader. Functionally, SDMI is trying to move on by shifting its focus beyond portable music players to such devices as mobile phones, radios, and voice recorders — all devices that could play a part in the distribution of pirated music but have yet to do so.

What happened to impede SDMI's progress? Sources say that the main problem was fundamental conflicts of interest between the three constituencies: record companies, consumer device makers, and technology vendors. The latter, which included vendors of various flavors of DRM technology, wanted the specifications of their solutions to be endorsed as the standard so that they could sell their products immediately as "SDMI-compliant" and their competitors would have to retool their offerings. The resulting squabbles among technology vendors — however typical in technology standards initiatives — proved detrimental to progress.

Consumer electronics makers, meanwhile, did not want to be forced to produce devices that limited the types of music files that consumers could play; this would limit their appeal. The dichotomy of Phase I and Phase II enabled them to get a product to market quickly, but would have required them to put out two lines of products and, worse, would have required consumers to upgrade. Record companies had the opposite goal of the consumer electronics makers: They *did* want to limit music players' ability to play files under certain conditions.

The result was a virtual logjam. Each component of the industry had too much invested in the status quo to want to compromise.

Other recent events in the music industry put SDMI's future further in doubt. With important court decisions coming down in favor of the record labels, the shrinking market caps of many online music ventures, and the deals that the record labels have been doing with companies such as Napster, MP3.com, and myplay, the balance is shifting back to the record labels. They aren't particular about the need to use technology to protect their franchise; if they can get enough help through the legal system or the financial markets, they may no longer see the need for a technological solution.

Be that as it may, the record labels are taking matters into their own hands by launching subscription services that are based on existing DRM technology from Microsoft and RealNetworks: PressPlay, a joint venture of Sony and Universal, and MusicNet, which includes EMI, BMG, and Warners (the other three major labels). We discuss these in Chapters 2 and 7. It is possible that PressPlay and MusicNet will raise the hackles of antitrust regulators for reasons related to those stated in the section "The Role of Open Standards in DRM," earlier in this chapter. But otherwise, we believe that PressPlay and MusicNet represent the future – insofar as there is one – for online digital music.

Other Standards

We conclude this chapter with brief looks at a couple of other standards initiatives that are related to DRM, but which we believe will have less impact than the ones we have discussed in detail thus far.

eXtensible Media Commerce Language

The eXtensible Media Commerce Language (XMCL) was announced in June 2001 by RealNetworks, the vendor of streaming media technology, at the Streaming Media West trade show in Los Angeles. A Web site for the standard exists (`www.xmcl.org`), as does a draft specification (`www.xmcl.org/specification.html`).

XMCL, like XrML and ICE, is a technology for specifying content rights information based on the XML metalanguage. It is meant for describing "business rules" that govern access to content and the consideration for which that access is obtained – that is, rights models, as described in Chapter 4. XMCL is meant to be a standard language for communicating between DRM solutions and various systems, including e-commerce storefronts, payment systems, Web publishing, digital asset management, and customer relationship management systems. Although it was invented in the context of streaming media, XMCL was designed to apply to a wide range of content types and commerce models.

The most important top-level entity in an XMCL specification is the `license`. License specifications contain the following:

- Content metadata, including identification and keywords.
- A specification of the period during which the license is valid.
- Specifications of usage (render) rights, as well as copy and transfer (transport) rights, including rights extents.

The specification shows a compact design that is geared toward efficient implementation rather than comprehensiveness. It was presumably designed with RealNetworks' Media Commerce Suite in mind.

CROSS REFERENCE: See Chapter 7 for more on Media Commerce Suite.

From a purely technological standpoint, XMCL is a standard with its heart in the right place. It addresses the needs of technologists who have found XrML too heavyweight to implement and who are put off by the fact that an individual company, ContentGuard, holds patents to XrML. But more to the point, XMCL has been advanced by vendors who feel that XrML is really Microsoft's technology: As we point out in the section on XrML, earlier in this chapter, Microsoft is really the only vendor that has implemented XrML so far, although it's expressed its intention to advance it as an open standard.

RealNetworks has clearly been trying to rally companies around XMCL as an anti-Microsoft standard. A long list of companies became signatories to the press release that announced XMCL, including content providers (America Online, Bertelsmann, Clear Channel, EMI, MGM, Sony Pictures Digital Entertainment, and Starz Encore Group), technology vendors (Adobe, Artesia, Avid, eMotion, IBM, InterTrust, Rightsline, Sun Microsystems, and Virage), and various others. This makes XMCL yet another in the equally long list of "everyone against Microsoft" initiatives that have regularly appeared in the technology industry over the past years.

Unfortunately, very few such initiatives pan out. Of all the preceding press release signatories, we know of none that is actually working on its own implementations of XMCL, whether for in-house use or for its own product lines. A few that we talked to admitted that they contributed to the press release because they liked the *idea* of such a standard, not because they actually intended to devote any resources to implementing it.

Although the XMCL working group has stated its intentions to submit the specification to the World Wide Web Consortium (W3C) for ratification as an official Internet standard, we see XMCL losing what little momentum it had. At this writing, no updates have been posted to the original draft specification (dated June 19, 2001), no related product announcements have been made, and our attempts to get further information from the people responsible for XMCL at RealNetworks have not been productive. We suspect that XMCL will go the way of many well-intentioned initiatives in this ever-changing industry.

Open Digital Rights Language

The Open Digital Rights Language (ODRL) is the brainchild of Renato Iannella of IPR Systems, a vendor of digital asset management technology located in Sydney, Australia. Like XMCL, it is an XML-based language for expressing rights specifications that stays clear of implementation and rights enforcement issues.

ODRL contains an elegant rights-modeling language with components that map closely to the rights model elements discussed in Chapter 4:

- *Permissions*, including Usage, Reuse, and Transfer, which correspond roughly to render, derivative work, and transport rights as described in Chapter 4.
- *Constraints*, which correspond to the rights extents of Chapter 4.
- *Requirements*, which map to consideration in Chapter 4.

In addition, ODRL contains components that model rightsholders, their royalty obligations, and their agreements with other parties.

ODRL is more comprehensive in its rights modeling power than XMCL. In fact, it resembles a lighter-weight version of XrML, without some of XrML's features for security levels and other implementation-level concerns. It was clearly influenced by XrML's predecessor, DPRL, as well as by the massive metadata description language <*indecs*>.

CROSS REFERENCE: See the section "Discovery metadata" earlier in this chapter for information on discovery metadata standards.

There is also an ODRL Web site (`www.odrl.net`) and a draft specification (`www.odrl.net/0.9/ODRL-09.pdf`) dated July 2001. However, ODRL appears to have gotten no traction outside of Australia and the Pacific Rim. Sponsors of the effort, in addition to IPR Systems, include two Australian intellectual property law firms, an E-book retail site that IPR built, and a Korean DRM startup vendor called ARPA (no relation to the Advanced Research Projects Agency of the U.S. Defense Department).

ODRL has been cited as influential by several organizations, including Hewlett-Packard Labs, the Open E-Book Forum, and Andersen Consulting (now Accenture), the latter in its study on eBooks for the Association of American Publishers. However, we do not expect ODRL to advance far beyond that in importance to the market.

World Wide Web Consortium

We conclude this section with a look at what the World Wide Web Consortium (W3C, `www.w3.org`) is doing about DRM. The W3C is, of course, the original governing body of the Web (insofar as any one organization could be said to govern it) and the ultimate arbiter of Web technology standards. DRM's ascent beyond the realm of publishers and other media companies into the mainstream of the Internet, which we believe is inevitable, is much more likely to happen through open standards if the W3C espouses them.

At this writing, several vendors and standards bodies have made presentations to the W3C or the unaffiliated Internet Engineering Task Force (IETF) to get them to endorse various DRM technologies as Internet standards. So far, nothing definitive has happened. The W3C held a workshop on DRM for the Web in Sophia Antipolis, the technology center in the south of France, in January 2001. People who delivered position papers at the workshop included representatives from DRM technology vendors, content providers, major technology firms, and standards bodies, as well as consultants and academics. Significantly, although most of the position papers espoused approaches that reflected their authors' own technologies, few of them took the position that DRM was antithetical to the original spirit of the Internet and, if espoused by the W3C, would induce a consumer backlash.

Since then, Renato Iannella of IPR Systems in Australia, author of the ODRL specification (see previous discussion), has been trying to gather support for a W3C DRM interest group. It remains to be seen how much momentum can be achieved. There is a fundamental conflict among people active in the W3C about whether DRM should even be considered a legitimate area of standardization at all. If that attitude prevails within the W3C, DRM standardization will fall to the narrower community of DRM technology vendors and content providers.

One thing is for sure: Standards are crucial. That is why we chose to go into so much detail about them in this book and why two of us have devoted significant chunks of our careers to working on them. We have more to say about the future and destiny of DRM standards in the final chapter of this book.