A Metadata Kernel for Electronic Permanence

John A. Kunze
University of California
jak@ckm.ucsf.edu

Abstract

This paper presents a streamlined metadata record format designed to support the permanence of network discoverable objects. It starts with the Dublin Core consensus and distills out a subset of four semantic buckets − a metadata kernel − that balances the needs for adequate identification of persistent objects and for low cost metadata generation. To minimize the burden of creating, understanding, and manipulating data in those buckets, a very simple record format has been designed, called an Electronic Resource Citation (ERC). The basic ERC can be parsed by two lines of Perl code. Beyond permanence support, the ERC design suggests quite a new path for the ongoing development of simple metadata; readers familiar with the current evolutionary challenges may find the ERC to be simpler, and yet more complete, compact, extensible, and international than the Dublin Core.

Keywords: *persistent identifier, permanence, simple metadata, stub records, preservation*

1. Permanence of Electronic Objects

Permanence of electronic information, namely, the extent to which structured digital data remains predictably available through known channels, is a central concern for most organizations whose mission includes an archival function. As providers of information resources have tended to focus their efforts on getting information up and available on the internet, understanding of the issues related to keeping it available as part of the cultural and historical record has lagged considerably behind. The result has been not only a general lack of confidence in permanent electronic reference on the internet, but also in a reluctance on the part of authors and publishers to rely on the internet as a publishing medium.

Work in progress concerning electronic permanence suggests that permanence of electronic information − indeed, of any information, whether physical, abstract, or digital − is closely tied to persistent identification and to certain kinds of metadata. The recently proposed Archival Resource Key (ARK) [1] is a naming scheme designed to support long term durable references to objects. Here, the term "object" means the same thing as information resource.

2. Metadata for Persistent Identification

Schemes for persistent identification of network-accessible objects are not new. Since the early 1990's, a series of naming systems have been proposed to support persistent identification, the better known of these being the Uniform Resource Name [2], the Digital Object Identifier [3], and the Persistent URL [4]. Each of the schemes employed some sort of recognizable character sequence in the identifier (such as the prefix "urn:") in order to alert the recipient to the name assigning authority's (the organization that minted the identifier) intention to make it persist forever.

A founding principle of the ARK scheme is that persistence is purely a matter of service. Persistence is neither inherent in an object nor conferred on it by a particular naming syntax or prefix, not even by the prefix "ark:" used in the ARK scheme. Rather, it is achieved through a provider's successful stewardship of objects and their identifiers. No one can tell if successful stewardship will take place because no one can predict the future. Reasonable conjecture, however, may be based on a specific promise from a provider with a known reputation.

The ARK is an identifier that binds an object together with metadata conveying not only a commitment statement from an identified provider but also an object description, the latter being required to furnish complete identification. Although it is not uncommon in the online publishing world to see URLs or URL fragments doubling as ad hoc metadata containers, as in the case of

```
ajpcell.physiology.org/cgi/content/full/278/2/C391
```

such identifiers do not age or travel well. Veteran producers and providers who wish to create a long term name tend to use an *opaque* identifier, such as ISBN 0-201-03803-X. Such an identifier contains little if any widely recognizable natural language meaning because natural language is unstable (e.g., compare the meaning of "gay" in 1998 and 1958) and because inclusion of language and character set tags needed to properly specify the identifier's semantic context would make it unwieldy. The appetite for

this sort of URL-cum-metadata-container can be satisfied without sacrificing persistence, by having as a fundamental requirement that metadata be tightly bound to the ARK identifier.

As mentioned, ARK metadata is also required in order to convey provider promises (or, more generally, policies). Such promises are multi-faceted covenants that bind the word of a provider to a specific commitment. A realistic commitment is not a simple binary statement (persistent or impersistent), but details independently varying dimensions of persistence, such as whether an identifier will be reassigned, whether the object will be available for the provider's lifetime, and how the object content may change (e.g., invariant, subject to correction, subject to revision) [5]. A machine readable statement about object support is a kind of object metadata.

The final requirement of an ARK is access, where feasible. While ARK access does not deal directly with metadata, a credible promise of access implies a commitment to the identifier and to the metadata that the committing organization needs, if only to manage its archival store. That credibility is ultimately up to the recipient of an ARK to decide. Given an ARK, anyone can examine the metadata containing the promise and record for future reference the metadata containing the description. These are the basic tools of permanence for network-discoverable objects.

3. Dublin Core and General Metadata

There are many possible protocols and record formats to service ARKs, but there is one especially simple and general format called the Electronic Resource Citation (ERC) that meets the demands of realistic permanence services. The Dublin Core (DC) metadata element set falls short of these demands because it fails to define (a) what a record is, (b) what elements a minimal object description is required to have, (c) in what manner certain fundamental elements such as Author and Date must be laid out, (d) what element to use to indicate metadata provenance (the first step towards establishing its credibility), (e) a specific way to formally and informally extend DC elements with local or community-defined elements, (f) a specific way to specify internationally recognizable elements using language-neutral tags, and (g) what element to use to indicate a provider's support commitment.

The DC addresses some of these issues, but only by referring readers to a half-dozen unfinished, non-standard specifications that, in turn, rely on unfinished standards (e.g., RDF), and by asking communities to create still more documents to add their local flavors of DC to the mix (these are called *application profiles*). Such complexity is not only daunting to implementors, but also slows the formation of consensus on unresolved foundation matters that affect all DC users. The ERC offers a way to capture the valuable DC consensus and move it forward in a context of austere design simplicity.

While object discovery was the original orientation of the Dublin Core, and while ERC metadata is quite general, the ERC semantics described here tend to have the flavor of metadata for archival and access management. Of course, any metadata is potentially useful for discovery by certain target audiences, but ERC metadata was not designed specifically with general internet discovery in mind. Still, to the extent that users and index builders are attuned, the ERC format imposes more regularity than ordinary Dublin Core; the expectation is that the ERC should stand a reasonable chance of meeting or exceeding the Dublin Core in support of precise and comprehensive database search results.

Permanence does not necessarily circumscribe the general utility of ERC metadata. As global experience with metadata (DC, FGDC, INDECS, etc.) is gained and the costs reckoned, human-created metadata may prove too expensive except for the highest priority objects, often the very objects for which providers receive preservation funding. Preservation is a methodical and perpetually underfunded business requiring the low-cost creation of archival management metadata. To the extent that general metadata finds much of its human expression in the context of permanence, the ultra simple, permanence-minded ERC format looks more like a general solution. Implementation experience with ERCs is currently limited to a prototype ARK service (http://ark.nlm.nih.gov/) at the US National Library of Medicine.

4. ERC Overview

An Electronic Resource Citation (ERC) is a simple, compact, and printable record for holding data associated with an information resource. By design, the ERC is a metadata format that balances the needs for expressive power, very simple machine processing, and direct human manipulation. Rather than encourage an unhealthy dependence on complex software for routine record handling, the ERC places paramount importance on ease of system implementation and human training; the syntax and semantics must be deterministic and readily learned. The remainder of this paper describes the ERC record and element syntax, and the semantics in support of permanence as derived from the DC.

A founding principle of the ERC is that direct human contact with metadata will be a necessary and sufficient condition for the near term rapid development of metadata standards, systems, and services. Thus the machine-processable ERC format must only minimally strain people's ability to read, understand, change, and transmit ERCs without their

relying on intermediation with specialized software tools. The basic ERC needs to be succinct, transparent, and trivially parseable by software. For example, the following ERC is parseable with two lines of Perl scripting instructions:

```
erc:
who:    Lederberg, Joshua
what:   Studies of Human Families for
              Genetic Linkage
when:   1974
where:  http://profiles.nlm.nih.gov/BB/AA/TT/tt.pdf
```

In the current Internet, it is natural to seriously consider using the Extensible Markup Language (XML) [6] as an exchange format because of predictions that it will obviate many ad hoc formats and programs, and unify much of the world's information under one reliable data structuring discipline that is easy to generate, verify, parse, and render. It appears that for metadata encoding, however, XML is still only catching on after years of standards work and implementation experience. The reasons for this are unclear, but for now very simple XML interpretation is still out of reach. At least one important caution is that XML structures are tiring to the eyes, taking up an amount of display (and page) space that significantly exceeds that of traditional formats.

Borrowing instead from the data structuring format that underlies the epidemic spread of email and web services, this first format for supporting ARK requirements (the ERC) is based on email and HTTP headers (RFC822) [7]. There is a naturalness to the label-colon-value format that barely needs explanation to a person beginning to enter ERC metadata. Moreover, the format, or one very like it, is routinely relied upon when Dublin Core experts really require simple metadata, namely, when discussing metadata concepts with funding sources and when a standards discussion reaches an impasse that can only be breached by absolute clarity of representation. This is a powerful empirical statement. Such simplicity does come at a cost, however; compared to XML, the ERC format is weak when it comes to nesting records more than a few levels deep. Application builders should consider this in choosing a record format.

5. ERC Element Semantics

The Electronic Resource Citation design starts with the observation that the Dublin Core's primary contribution is the international, interdisciplinary consensus that identified fifteen semantic buckets (element categories), regardless of how they are labeled. The fifteen buckets are trimmed down and recombined to isolate a cross-domain subset of four derived elements, called ERC kernel metadata. To support permanence across object types, the kernel is designed for general cross-domain description.

Kernel metadata comes with strict compliance rules, but any number of extra non-kernel metadata elements can be appended to enrich the record once the four cross-domain kernel elements have been specified. In particular the ERC requires, in exactly this order, the following four basic elements:

who − a responsible person or party
what − a name or other human-oriented identifier
when − a date important in the object's lifecycle
where − a location or machine-oriented identifier

To a first approximation, the ERC cross-domain kernel pretends that every object in the universe can have a uniform minimal description, that it doesn't matter what type of object it is, or whether one reads it, interacts with it, smokes it, wears it, or navigates it. Of course, such uniformity of description for some object types requires more semantic sacrifice than for others, but any loss due to the approximation can be mitigated by appending other elements. Thus the ERC permits a semantically rich and nuanced description to co-exist in a record along with a basic cross-domain description. Both sophisticated and naive recipients of such a record can extract the level of meaning from it that best suits their respective needs and abilities. Key to unlocking the richer description is a controlled vocabulary of ERC record types (not explained here) that permit knowledgeable recipients to apply defined sets of additional assumptions to the record.

To support internationalization and accurate, long term references for element categories, the ERC requires that each semantic unit (element, qualifier, or other controlled value) have not only a standard, human-oriented label but also a language-neutral, semi-numeric *concept identifier*. In the following ERC, the familiar four elements are again present in the required order, but the element labels are non-standard (in fact they are in German). Concept identifiers are given in parentheses, and a fifth element has also been added.

```
erc:
wer(h1):   Miller, Alice
was(h2):   Am Anfang war Erziehung
wann(h3):  1983
wo(h4):    http://www.amazon.com/exec/obidos/ASIN%{
             /0374522693/thenaturalchildp %}
Titel(h89): (en) For your Own Good:  Hidden Cruelty
        in Child-Rearing and the Roots of Violence
```

Either the element label or the identifier may appear, with the identifier taking precedence if both are present. A feature of this precedence rule is that non-standard element labels appearing together with concept identifiers (as above) may be used for the convenience of regional metadata providers and their users, while the same elements can be exported globally without loss of internationally recognized semantics. ERCs at large can be received by services that provide appropriate display translations of known tags (concept identifiers and standard labels)

and take alternative action for unknown tags, such as suppressing them in summary listings.

6. ERC Syntax

The next example shows five text lines comprising a simple Electronic Resource Citation. It is a sequence of five metadata elements ending in a blank line. An element consists of a label, a colon, and an optional value.

```
erc:
who:   Gibbon, Edward
what:  The Decline and Fall of the Roman Empire
when:  1781
where: http://www.ccel.org/g/gibbon/decline/
```

A long value may be folded (continued) onto the next line by inserting a newline and indenting the next line. A value can be thus folded across multiple lines. Here are two example elements, each folded across four lines.

```
who/created: University of California San Francisco,
     AIDS Program at San Francisco General Hospital
     | University of California, San Francisco,
     Center for AIDS Prevention Studies
what/Topic:
      Heart Attack
      | Heart
            Failure
```

An element value folded across several lines is treated as if the lines were joined together on one long line. For example, the second element from the previous example is considered equivalent to

```
what/Topic: Heart Attack | Heart Failure
```

An element value may contain multiple values, each separated from the next by a '|' (pipe) character. The above element contains two values.

For annotation purposes, any line beginning with a '**#**' (hash) character is treated as if it were not present; this is a "comment" line (a feature not available in email or HTTP headers). For example, the following element is spread across four lines and contains two values:

```
what/Topic:
   Heart Attack
#  | Heart Failure  -- hold until next review cycle
   | Heart Diseases
```

That's basic syntax. Overall record layout is next.

6.1. The ERC Notion of *Story*

An ERC record is organized into one or more distinct *segments*, where each segment is a list of elements headed by a label beginning with the letters "**erc**". A segment boundary occurs whenever such a segment label is encountered. Every segment tells a *story* about a different aspect of the information object according to the story's type, and the label reveals that type. Types include the story of the object's expression, of its provenance, of its content, etc. A story is thus a sequence of elements, each making an assertion relevant to the type of story.

The basic label "**erc:**" introduces the story of an object's expression (e.g., its publication, installation, performance, or discovery). The label "**erc-about:**" introduces the story of an object's content − what it is about − and contains elements for things like subject, description, location, and time period covered, etc. A written work available in several combinations of language and format can have an ERC that includes several segments labeled "**erc-obj:**", each of which describes the low-level characteristics and locations of the variant forms. Elements appearing in such stories are familiar to users of the Dublin Core.

Other labels introduce stories with elements having no counterparts in the Dublin Core, but which are nonetheless vital to electronic permanence. The label, "**erc-support:**", introduces the story of a support commitment made to an object, and is critical to a user's ability to forecast the persistence of an object and its identifier. A segment labeled "**erc-from:**" contains elements telling the story of the ERC's origin. Without some indication of provenance, the credibility of a random metadata record cannot be judged, especially when it appears in a record set aggregated from a large number of unsecured channels − precisely how all internet search engine crawlers gather records. This basic ERC support for provenance affects the perceived value of records for all users, not just those interested in permanence. The Dublin Core does not define a way to label records as to their origin (in fact, it does not define what a record is), so providers of high quality metadata must turn elsewhere to make their metadata assertions distinguishable from spam.

From an earlier example, here is an ERC with two segments.

```
erc:
who:   Lederberg, Joshua
what:  Studies of Human Families for Genetic Linkage
when:  1974
where: http://profiles.nlm.nih.gov/BB/AA/TT/tt.pdf
erc-support:
who:   NIH/NLM/LHNCBC
what:  Permanent, Unchanging Content
# Note to ops staff:  date needs verification.
when:  2001 04 21
where: http://ark.nlm.nih.gov/yy22948
```

Segment stories are told according to journalistic tradition. While any number of pertinent elements may appear in a segment, priority is placed on answering the questions who, what, when, and where at the beginning of each segment so that readers can make the most important selection or rejection decisions as soon as possible. To keep things simple, the listed ordering of the questions is maintained in each segment (as it happens most people who have been

exposed to this story telling technique are already familiar with the above ordering).

The four questions are answered by using corresponding element labels. The four element labels can be re-used in each story segment, but their meaning changes depending on the segment label (the story type) under which they appear. In the example above, "**who**" is first used to name a document's author, and subsequently used to name the permanence guarantor (provider). Similarly, "**when**" first lists the date of object creation, and in the next segment lists the date of a commitment decision. Four labels appearing across three segments effectively map to twelve semantically distinct elements. Distinct element meanings are mapped to Dublin Core elements in a later section.

6.2. The ERC Anchoring Story

Each ERC contains an *anchoring story*. It is usually the first segment labeled "**erc:**" and it concerns an "anchoring" expression of the object. An *anchoring expression* is the one that a provider deemed the most suitable basic referent given the audience and application for which it produced the ERC. If it sounds like the provider has great latitude in choosing its anchoring expression, it is because it does. A typical anchoring story in an ERC for a born-digital document would be the story of the document's release on a web site; such a document would then be the anchoring expression.

An anchoring story need not be the central descriptive goal of an ERC record. For example, a museum provider may create an ERC for a digitized photograph of a painting but choose to anchor it in the story of the original painting instead of the story of the electronic likeness; although the ERC may through other segments prove to be centrally concerned with describing the electronic likeness, the provider may have chosen this particular anchoring story in order to make the ERC visible in a way that is most natural to patrons (who would find the Mona Lisa under da Vinci sooner than they would find it under the name of the person who snapped the photograph or scanned the image). In another example, a provider who creates an ERC for a dramatic play as an abstract work has the task of describing a piece of intangible intellectual property. To anchor this abstract object in the concrete world, if only through a derivative expression, it makes sense for the provider to choose a suitable printed edition of the play as the anchoring object expression (for the anchoring story to describe) of the ERC.

The anchoring story has special rules designed to keep ERC processing simple and predictable. Each of the four basic elements (who, what, when, and where) must be present, unless a best effort to supply it fails. In the event of failure, the element label still appears, but a special code (described later) is used to explain the missing value. As before, the four elements must appear at the beginning of the segment and may only be used in the prescribed order. A minimal ERC would normally consist of just an anchoring story and the element quartet, as illustrated in the next example.

```
erc:
who:   National Research Council
what:  The Digital Dilemma
when:  2000
where: http://books.nap.edu/html/digital%5Fdilemma
```

A minimal ERC can be abbreviated so that it resembles a traditional compact bibliographic citation that is nonetheless completely machine processable. The required elements and ordering make it possible to eliminate the element labels, as shown here.

```
erc: National Research Council
     | The Digital Dilemma | 2000
     | http://books.nap.edu/html/digital%5Fdilemma
```

6.3. ERC Elements

As mentioned, the four basic ERC elements (who, what, when, and where) take on different specific meanings depending on the story segment in which they are used. By appearing in each segment, albeit in different guises, the four elements serve as a kind of checklist and valuable mnemonic device to help in constructing minimal story segments from scratch. Again, it is only in the anchoring story segment that all four elements are mandatory.

Here are some mappings between ERC elements and Dublin Core elements [8].

Element	Equivalent DC Element
erc	
`who(h1)`	Creator/Contributor/Publisher
`what(h2)`	Title
`when(h3)`	Date
`where(h1)`	Identifier
erc-about	
`who(h11)`	*none*
`what(h12)`	Subject
`when(h13)`	Coverage (temporal)
`where(h14)`	Coverage (spatial)

The basic element labels may also be qualified to add nuances to the semantic categories that they identify. Elements are qualified by appending a '`/`' (slash) and a qualifier term. Often qualifier terms appear as the past tense form of a verb because it makes re-using qualifiers among elements easier.

```
who/published:  ...
when/published: ...
where/published: ...
```

Using past-tense verbs for qualifiers also reminds providers and recipients that element values contain

transient assertions that may have been true once, but that tend to become less true over time. Recipients that don't understand the meaning of a qualifier can fall back onto the semantic category (bucket) designated by the unqualified element label. Inevitably recipients (people and software) will have diverse abilities in comprehending elements and qualifiers.

Any number of other elements and qualifiers may be used in conjunction with the quartet of basic segment questions. The only semantic requirement is that they pertain to the segment's story. Also, it is only the four basic elements that change meaning depending on their segment context. All other elements have meaning independent of the segment in which they appear. If an element label stripped of its qualifier is still not recognized by the recipient, a second fall back position is to ignore it and rely on the four basic elements.

Elements may be either Canonical, Provisional, or Local. Canonical elements constitute the metadata *vernacular*, and are officially recognized via a registry. All elements, qualifiers, and segment labels used in this paper up until now are Canonical or Provisional. Provisional elements are also officially recognized via the registry, but have only been proposed for inclusion in the vernacular; they are distinguished by the first character, which must be an upper case letter. To be promoted to the vernacular, a provisional element passes through a vetting process during which its documentation must be in order and its community acceptance demonstrated. Local elements are any elements not officially recognized in the registry.

```
erc:
who: Bullock, TH | Achimowicz, JZ | Duckrow, RB
        | Spencer, SS | Iragui-Madoz, VJ
what: Bicoherence of intracranial EEG in sleep,
        wakefulness and seizures
when: 1997 12 00
where: http://cogprints.soton.ac.uk/%{
        documents/disk0/00/00/01/22/index.html %}
in: EEG Clin Neurophysiol | 1997 12 00
        | v103, i6, p661-678
IDcode: cog00000122
# ---- new segment ----
erc-about:
what/_subcategory: Bispectrum | Nonlinearity
    | Epilepsy | Cooperativity | Subdural
    | Hippocampus | Higher moment
# ---- new segment ----
erc-from:
who: NIH/NLM/NCBI
what: pm9546494
when/Reviewed: 1998 04 18 021600
where: http://ark.nlm.nih.gov/12025/pm9546494?
```

In the three-segment example ERC above, local elements are immediately distinguishable because they begin with an underscore ('_'). All such terms are reserved for spontaneous provider use in local names for elements and qualifiers. Any recipient of external ERCs containing such terms will understand them to be part of the originating provider's local metadata dialect.

The example includes one provisional element (**IDcode**), one local qualifier (**_subcategory**), and one provisional qualifier (**Reviewed**). Also, the segment boundaries have been emphasized by comment lines. As such, they are ignored by automated processors, but remain at the service of those who maintain or teach about metadata.

6.4. ERC Element Values

ERC element values tend to be straightforward strings. If the provider intends something special for an element, it will so indicate with markers at the beginning of its value string. The markers are designed to be uncommon enough that they would not likely occur in normal data except by deliberate intent. Markers can only occur near the beginning of a string, and once any octet of non-marker data has been encountered, no further marker processing is done for the element value. In the absence of markers the string is considered pure data; this has been the case in all the examples seen thus far. An element with all three optional markers in place looks like this.

```
who:[g=es](:LC), Congreso (EE.UU.), Biblioteca del
```

The general form of such an element's value is

[*markup-flags***]** **(:***ccode***)** **,** DATA

where "*ccode*" (a control code) and "*markup-flags*" can assume different values. In processing, the first non-whitespace character of an ERC element value is examined. An initial '**[**' is reserved to introduce a bracketed set of markup flags (not described in this paper) that ends with '**]**'. If ERC data is machine-generated, each value string may be preceded by "**[]**" to prevent any of its data from being mistaken for markup flags. Once past the optional markup, the remaining value may optionally begin with a controlled code. A controlled code always has the form "**(:***ccode***)**", for example,

```
who: (:unkn) Anonymous
what: (:791) Bee Stings
```

Any string after such a code is taken to be an uncontrolled (e.g., natural language) equivalent. The code "**unkn**" indicates a conventional explanation for a missing value (stating that the value is unknown). The remainder of the string makes an equivalent statement in a form that the provider deemed most suitable to its (probably English-speaking) audience. The code "**791**" could be a fixed numeric topic identifier within an unspecified topic vocabulary. Any such code may be ignored by those that do not understand it. There are several codes to explain different ways in which a required element's value may go missing.

Missing value codes and their meanings.

`(:unkn)`	unknown (e.g., Anonymous, Inconnue)
`(:unav)`	value unavailable indefinitely
`(:unac)`	temporarily inaccessible
`(:unap)`	not applicable, makes no sense
`(:unas)`	value unassigned (e.g., Untitled)
`(:none)`	never had a value, never will
`(:null)`	explicitly empty
`(:unal)`	unallowed, suppressed intentionally
`(:tba)`	to be assigned or announced later

Once past an optional controlled code, the remaining string value is subjected to one final test. If the first next non-whitespace character is a '`,`' (comma), it indicates that the string value is "sort-friendly". This means that the value is (a) laid out with an inverted word order convenient for sorting items having comparably laid out element values (items might be the containing ERC records) and (b) that the value may contain other commas that indicate inversion points should it become necessary to recover the value in natural word order. This feature can be used to express Western-style personal names in family-name-given-name order. It can also be used wherever natural word order might make sorting tricky, such as when data contains titles or corporate names. Here are some example elements.

```
who:, van Gogh, Vincent
who:, Howell, III, PhD, 1922-1987, Thurston
who: , Acme Rocket Factory, Inc., The
who:, Mao Tse Tung
who: ,McCartney, Paul, Sir,
what:, Health and Human Services, United States
        Government Department of, The,
```

There are rules, not fully specified here, to use in recovering a copy of the value in natural word order if desired. The above example strings have the following natural word order values, respectively.

```
Vincent van Gogh
Thurston Howell, III, PhD, 1922-1987
The Acme Rocket Factory, Inc.
Mao Tse Tung
Sir Paul McCartney
The United States Government Department
    of Health and Human Services
```

The optional use of an initial comma to indicate the presence of sort-friendly values applies uniformly across all elements.

6.5. ERC Element Encoding and Dates

Some characters that need to appear in ERC element values might conflict with special characters used for structuring ERCs, so there needs to be a way to include literal characters that are protected from special interpretation. This is accomplished through an encoding mechanism that resembles the %-encoding familiar to handlers of URLs [9].

The ERC encoding mechanism also uses '`%`', but instead of taking two following hexadecimal digits, it takes one non-alphanumeric character or two alphabetic characters that cannot be mistaken for hex digits. It is designed not to be confused with normal web-style %-encoding. In particular decodes without risking unintended decoding of normal %-encoded data (which would introduce errors).

Some extension codes and their meanings.

`%!`	decodes to the element separator '`|`'
`%%`	decodes to a percent sign '`%`'
`%.`	decodes to a comma '`,`'
`%dq`	decodes to a double-quote '`"`'
`%_`	a *non-character* used as syntax shim
`%{`	a non-char. that begins an expansion block
`%}`	a non-char. that ends an expansion block

One particularly useful construct in ERC element values is the pair of special encoding markers ("`%{`" and "`%}`") that indicates an *expansion* block. Whatever string of characters they enclose will be treated as if none of the contained whitespace (SPACEs, TABs, Newlines) were present. This comes in handy for writing long, multi-part URLs in a way that is easy for both people and machines to read. For example, the locator in

```
where: http://foo.bar.org/node%{
            ? db = foo
            & start = 1
            & ranking = 5
            & buf = 2
            & query = foo + bar + zaf
       %}
```

can be reliably decoded into an equivalent URL; like many such URLs, however, it cannot be represented intact due to line length limitations, and readers are often left with printed line fragments such as

```
where: http://foo.bar.org/node?db=foo&start=1&rank-
ing=5&buf=2&query=foo+bar+zaf
```

to be re-assembled into a complete URL through unreliable guesswork.

In a parting word about element values, dates and times are commonly recurring types. ERC dates take on one of the following forms:

`1999`	(four digit year)
`2000 12 29`	(year, month, day)
`2000 12 29 235955`	(year, month, day, hr, min, sec)

All internal whitespace is squeezed out to get a normalized date suitable for lexical comparison and sorting. This means that the following dates,

`2000 12 29 235955`	(recommended for readability)
`2000 12 29 23 59 55`	
`20001229 23 59 55`	
`20001229235955`	(normalized date and time)

are all equivalent. Hyphens and commas are reserved to create date ranges and lists, for example,

`1996-2000`	(range of four years)
`1952, 1957, 1969`	(list of three years)
`1952, 1958-1967, 1985`	(mixed list of dates & ranges)
`20001229-20001231`	(range of three days)

Note that the Dublin Core's recommended date format [10] does not permit this natural and compact way of representing lists and ranges.

6.6. Stub Records and Internal Support

The ERC design introduces the concept of a *stub* record, which is an incomplete ERC intended to be supplemented with additional elements before being released as a standalone ERC record. A stub ERC record has no minimum required elements. It is just a group of elements that conforms to the ERC syntax but does not begin with "`erc:`". Two ERC stub records from an informal personal bibliography might look like this.

```
what:  good network security rag
where: www.counterpane.com/crypto-gram.html

what:  freedom through format filters
where: http://www.wvware.com/
```

ERC stubs are useful in internal record management processes where accuracy and timeliness is needed of elements that change frequently. To be ready for external use, however, an ERC stub must be transformed into a complete ERC record having the usual required elements. For example, an ERC stub can be used to hold metadata embedded in a document, where vital items such as URL, modification date, and size − which one would not omit from an export record − are nonetheless omitted from the stub simply because they are more robustly supplied by a computation at the time the export record is needed. A locally defined administrative procedure (not defined for ERC's in general) would effect the promotion of stubs into complete records.

While the ERC is a general-purpose container for exchange of resource descriptions, it does not dictate how records must be internally stored, laid out, or assembled by data providers or recipients. Arbitrary internal descriptive frameworks can support ERCs simply by mapping local records (e.g., on demand) to the ERC container format and making them available for export. Therefore, to support ERCs there is no need for a data provider to convert internal data to be stored in an ERC format. On the other hand, there is nothing to prevent any provider (such as one just getting started in resource description) from doing all its local business in the ERC format.

7. Conclusion and Future Work

The Electronic Resource Citation (ERC) is a general-purpose metadata container that can support the persistent object identification that is a prerequisite for electronic permanence. Its cross-domain element and qualifier vocabulary is sufficient to provide complete identification with a low-barrier entry point into basic object description. The ERC metadata structure is also capable of conveying which provider promised what level of permanence regarding an object, without which users have no basis for gauging the reliability of an identifier for long term reference.

Application builders requiring utter simplicity and determinism from metadata, independent of permanence support, will be interested in exploring the advantages offered by the ERC kernel's reformulation of the Dublin Core. These advantages suggest that in the evolution of simple metadata, strategies that embrace complexity (from the richness of XML/RDF, to the emerging diversity of new namespaces, profiles, schemas, and schemes) would be well-complemented by a parallel strategy (such as the ERC) that aggressively eschews complexity.

Acknowledgements

This work was made possible through the support of the US National Library of Medicine. The author is grateful for Jane Greenberg's helpful comments.

References

[1] J. Kunze, "The ARK Persistent Identifier Scheme", September 2001, http://www.ietf.org/ietf/draft-kunze-ark-02.txt.
[2] R. Moats, "URN Syntax", RFC 2141, May 1997.
[3] International DOI Foundation, "The Digital Object Identifier (DOI) System", February 2001, http://dx.doi.org/10.1000/203.
[4] K. Shafer, et al, "Introduction to Persistent Uniform Resource Locators", 1996, http://purl.oclc.org/OCLC/PURL/INET96.
[5] M. Byrnes, "Defining NLM's Commitment to the Permanence of Electronic Information", ARL 212:8-9, October 2000, http://www.arl.org/newsltr/212/nlm.html.
[6] Extensible Markup Language 1.0, October 2000, http://www.w3.org/TR/REC-xml.
[7] D. Crocker, "Standard for the format of ARPA Internet text messages", RFC 822, August 1982.
[8] Dublin Core Metadata Initiative, "Dublin Core Metadata Element Set, Version 1.1: Reference Description", July 1999, http://dublincore.org/documents/dces/.
[9] T. Berners-Lee, et al, "Uniform Resource Identifiers (URI): Generic Syntax", RFC 2396, August 1998.
[10] Date and Time Formats, W3C Note, September 1997, http://www.w3.org/TR/NOTE-datetime.