DCMI Kernel Metadata Community	J. Kunze
	A. Turner
	California Digital Library
	October 10, 2007

Kernel Metadata and Electronic Resource Citations (ERCs)

Abstract

Kernel metadata is a small prescriptive vocabulary designed to support highly uniform but minimal object descriptions for the purpose of orderly collection management. The Kernel vocabulary, based on a subset of the Dublin Core (DC) metadata element set, aims to describe objects of any form or category, but its reach is limited to a small number of fundamental questions such as who, what, when, and where. The Electronic Resource Citation (ERC), also specified in this document, is an object description that addresses those four questions using Kernel and other metadata elements.

Table of Contents

1. Goals of Kernel Metadata
2. The Kernel and the ERC in Context
3. Kernel Stories
   - 3.1. The Anchoring Story
   - 3.2. Story Summary
4. Kernel Summary and Dublin Core Crosswalk
5. The Kernel and the ERC
6. The ANVL/ERC Record Syntax
7. Kernel Label Structure
8. Kernel Sort-Friendly Values
   - 8.1. Initial Comma to Recover Natural Word Order
9. Kernel Value Structure
   - 9.1. Multiple Values and Subvalues
   - 9.2. Kernel Initial Value Conventions
   - 9.3. Special Kernel Standardized Value Codes
   - 9.4. Kernel Date Values
   - 9.5. Element Value Encoding
10. Kernel Changes New in this Specification (Sept 2007)
11. Vocabulary of Elements and Values
12. References
§ Authors' Addresses

---

1. Goals of Kernel Metadata

Kernel metadata is designed to assist orderly collection management by supporting the creation of brief but highly uniform object descriptions that can be listed, surveyed, and searched efficiently during normal collection maintenance and trouble-shooting activities. These descriptions serve as object surrogates that are convenient for automated sorting and filtering operations and are also eye-readable without specialized display software. The goal of Kernel metadata is to balance the needs for expressive power, very simple machine processing, and direct human manipulation of metadata records.

Kernel metadata is based on the Dublin Core (DC) metadata element set **[RFC5013]** maintained by the Dublin Core Metadata Initiative **[DCMI]**. Kernel elements are descriptors that identify various object properties. In principle they apply to any object in the universe, whether digital, physical, or abstract, following in the tradition of **[RFC3986]**. This extreme diversity of objects is approached with the hypothesis that highly variable and rich object descriptions can be directly comparable at the level of about four fundamental elements — who, what, when, and where — as applied to the *expression* of the object. This sequence is a recurring theme in the Kernel. In anticipation of future extensions to "how" and "why", we refer to the first four elements as "the four h's" (what they all have in common is an initial aspirated "h" sound, which is also shorter to say than "w").

Kernel-based descriptions make it possible to compare an extremely diverse set of objects. Comparison is possible even when many other elements co-exist with Kernel elements, or when a minor amount of information in other elements overlaps with Kernel element information. Regardless of whether an object is smoked, worn, navigated, or in any other way, interacted with, its Kernel based description ensures the presence of a few predictable points of commonality in the form of easily isolatable Kernel elements. Kernel elements provide a concise intersection of interoperable (or at least comparable) elements across a broad range of object descriptions.

---

2. The Kernel and the ERC in Context

The Kernel is a vocabulary of metadata elements, where an element pairs a label with a value. As a vocabulary, the Kernel offers but does not obligate the use of its terms. The Kernel specifies both metadata elements and how particular data values should be structured within the elements. These rules may be complemented by other conventions (e.g., **[AACR2]**), although this is not required. As with most vocabularies, ultimate responsibility for creating coherent and sensible descriptions lies with the metadata creator.

The Electronic Resource Citation (ERC) introduced in this document is a kind of object description that does obligate use of the four fundamental Kernel elements. Standard encoding methods such as **[RDF]** and **[XML]** may be used to format ERCs and Kernel metadata. It is also possible to encode modified forms of Kernel element values using other methods, such as **[MARC]** or **[MODS]**,

although some granularity of information may be lost in the process. One important user of Kernel metadata and ERC object descriptions is the **[ARK]** identifier scheme.

The practice of using non-Kernel elements along with Kernel elements is normal: Kernel elements may appear in the same record with metadata from other vocabularies, such as Dublin Core and **[PREMIS]**. The requirement to use the four fundamental Kernel elements (the four h's) at a minimum is imposed specifically in the context of a complete ERC record, such as,

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

The four h's provide an affordable set of comparable elements common to a wide range of divergent metadata and object types, but do so without limiting the expressive range of the records. The above description, however, is minimal and therefore limited to the story of an object's expression.

---

3. Kernel Stories

The Kernel has a concept of "story", which is an organizing principle that applies the questions of who, what, when, and where to different aspects of an object description. The four required Kernel elements address one particular aspect — the story of an object's expression — and in so doing form something similar to a traditional citation:

> *who* expressed it (from DC Creator, Contributor, and Publisher),
> *what* the expression was called (from DC Title),
> *when* it was expressed (from DC Date), and
> *where* the expression can be found (from DC Identifier).

One descriptive record may contain stories of different expressions of the same object, for example, its digital and physical expressions. Depending on the object type — article, photograph, dance, fossil — an object's expression could mean quite different things, such as its publication, installation, performance, or discovery. One descriptive record may also contain stories of several different types, such as what the object is about (its "aboutness"), the origin of the record itself, and the provider's organizational support for the object. More about these story types is given after first describing the story that anchors a descriptive record.

3.1. The Anchoring Story

Among all the stories that an object's descriptive record may contain, there is one that the provider deems the most suitable basic referent given its audience and application. This is called the "anchoring" story. The provider has great latitude in choosing its anchoring story, but usually it appears first in the record as a kind of object summary that can be easily isolated by the human eye (Kernel elements appearing anywhere in a record can always be easily isolated by automated processes). If the record contains only one story, the anchoring story is it, and the record consists of just the four h's. A typical anchoring story for a born-digital document would be the story of the document's release on a web site.

Digital objects that weren't born-digital often call for a slightly more subtle approach. The anchoring story is usually a convenient front door into a non-specialist experience of the object, and that typically means instant access, where possible, either to the object or to a reasonable facsimile. So for a physical object resulting from a creative act (a book, statue, photograph, etc), the first three of the four h's should be biased towards the story of the original act while the location of the expression should, if possible, be a machine-actionable identifier. Even though such an identifier leads to a derivative object, automated access is often deemed important enough to the initial experience to be included in the anchoring story.

The complete and pure stories of both the derivative and original objects can be told, if necessary, elsewhere in the record. Meanwhile, the chance to anchor the object description in a hybrid story that describes the original work but favors electronic access is consistent with the 'E' (for electronic) in ERC. Above, for example, a URL to the online version of the book written in 1781 is given in preference to its shelf location in a library.

An anchoring story need not be the central descriptive goal of an ERC (or any other) 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 stories 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 that 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 (to describe in the anchoring story) of the ERC.

3.2. Story Summary

This section contains the list of currently defined story types, with additional story types under development. As shown below, similarly named elements are used in the Kernel to address the stories of an object's content, its support, the provenance of the metadata record itself, etc. Only one story is required of a complete (non-stub) ERC, and only four of its elements must be present.

```
who:   a responsible person or party (required)
what:  a name or other human-oriented identifier (required)
when:  a date important in the object's lifecycle (required)
where: a location or system-oriented identifier (required)
```

```
how:   (under construction) a formal type designator
```

```
about-who:   a person or party figuring in the information content
about-what:  a subject or topic figuring in the information content
about-when:  a time period covered by the information content
about-where: a location or region covered by the information content
about-how:   a description of the information content
```

```
meta-who:   a person or party responsible for the record
meta-what:  a short form of the identifier for the record
meta-when:  the last modification date of the record
meta-where: a location of the fullest form of the record
```

```
support-who:   a person or party responsible for the object
support-what:  a short form of the commitment made to the object
support-when:  the last modification date of the commitment
support-where: a location of the fullest form of the commitment
```

---

4. Kernel Summary and Dublin Core Crosswalk

Each Kernel element label has a coded synonym (the SYN column below) that consists of the letter 'h' followed by a number, such as h1, h2, h3, etc. The following table, organized by "story", summarizes the rough correspondence between Kernel elements and Dublin Core elements; the vocabulary section of this document details the true correspondence and element restrictions.

```
STORY           KERNEL LABEL    SYN    DUBLIN CORE APPROXIMATION

erc:            *  who          h1     Creator/Contributor/Publisher
The story of    *  what         h2     Title
an object's     *  when         h3     Date
expression.     *  where        h4     Identifier (permanent)
                   how          h5     (reserved Type restriction**)

about-erc:       about-who      h11    Subject (personage)
The story of     about-what     h12    Subject
an object's      about-when     h13    Coverage (temporal)
content.         about-where    h14    Coverage (spatial)
                 about-how      h15    Description

support-erc:    support-who     h21    (no equivalent)
The story of    support-what    h22    (no equivalent)
an object's     support-when    h23    (no equivalent)
support.        support-where   h24    (no equivalent)

meta-erc:         meta-who      h31    (no equivalent)
The story of      meta-what     h32    (no equivalent)
this record's     meta-when     h33    (no equivalent)
expression.       meta-where    h34    (no equivalent)

*  A complete ERC requires a non-missing value for this element.
** Under development.
```

Where Kernel elements map to Dublin Core (DC) elements, the map is roughly one-to-one, but with a few notable exceptions.

1. "who" maps to DC Creator, but if no Creator use Publisher, and if no Publisher, use Contributor; "who" resembles what was once considered in DCMI to be an "agent" element
2. "about-when" maps to the temporal aspect of DC Coverage and "about-where" maps to the spatial aspect of DC Coverage.
3. The Kernel assumes that most values, especially personal names given in "who", will be given in "sort-friendly" manner, for example, "lastname, firstname" for western names and natural word order for Chinese names.
4. The Kernel assumes **[TEMPER]** format for dates in order to express date ranges, lists, approximate dates, and BC dates (not possible, for example, with **[W3CDTF]**).

---

5. The Kernel and the ERC

This table illustrates the strong connection between the story concept in the Kernel and the ERC. While the Kernel is a vocabulary, it is the ERC that brings the assumptions about required elements. An ERC that does not contain all four h's is still a useful container, as when a description is being constructed, but it is classified as a "stub ERC". In the case of a stub, such as,

```
erc:
what:   The Digital Dilemma
where:  http://books.nap.edu/html/digital%5Fdilemma
```

the "erc:" label indicates that Kernel vocabulary elements are expected, and later inspection discloses that this ERC is incomplete.

An abbreviated form of any story can be given using the story label as an element label, and then constructing one long value by listing each of the story elements' values, in the order shown above, separated by a solidus ("|"). Because this composite value drops the constituent value labels, the ordering must be strictly observed so that the corresponding elements can be accurately identified. The

abbreviated form of the example from section 2 is:

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

A story label appearing with no value may be useful in visually setting off a region of a record but otherwise has no significance. The one exception is that the "erc" label, with or without an accompanying value, serves as a kind of record label that declares an object description to be an ERC.

Any story label can introduce an abbreviated story form, such as,

```
meta-erc:  NLM | pm9546494 | 19980418
           | http://ark.nlm.nih.gov/12025/pm9546494??
about-erc:   | Bispectrum ; Nonlinearity ; Epilepsy
               ; Cooperativity ; Subdural ; Hippocampus
```

There is no general requirement concerning missing values for these story labels (as for the "erc" label). It is common for composite Kernel elements to be constructed with subelement ordering that echoes the familiar who, what, when, where pattern.

Future versions of the Kernel may extend the four h's with two additional but non-required elements: how and why. These element names are reserved but under construction.

---

6. The ANVL/ERC Record Syntax

One way to represent an ERC is to use ANVL (A Name-Value Language), a simple text-based record syntax in the tradition of classic internet protocols such as **[RFC2822]**. Here is an example of an ERC as an ANVL record:

```
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
note:   This is an arbitrary note inside a
        small descriptive record.
```

What makes this ANVL record a complete ERC record is the "erc:" label and the presence of the four required elements.

It should be possible to represent an ERC in many different encodings (e.g., XML with a specific schema), provided the Kernel rules for element labels and values are followed. The Kernel rules coincide with rules for ANVL labels and values. Because ANVL is concise and easy to read, we will continue to use it in examples throughout this document.

As an ANVL record, the ERC is a sequence of elements beginning with "erc::" and ending in a blank line (who newlines in a row). While the ERC will look different in other encodings, in ANVL,

1. The record begins with "erc:" and ends at the first blank line.
2. Each element consists of a label, a colon, and an optional value.
3. A long value may be folded (continued) onto the next line by inserting a newline and indenting the next line.
4. A line beginning with a number sign ("#") is to be treated by recipients as if it were not present (in programmer terms, this would be called a *comment* line).

A value can thus be folded across multiple lines. An element value folded across several lines is treated as if the lines were joined together on one long line; thus the "note" element above is considered the same as

```
note:   This is an arbitrary note inside a small descriptive record.
```

That is all that this document has to say about ANVL, a complete description of which is detailed in the ANVL specification **[ANVL]**.

Independent of ANVL or any other encoding, there are rules for encoding ERCs in any concrete syntax. Inside Kernel element labels and values these rules happen to coincide with the ANVL element rules. The basic features of any format holding Kernel elements are:

1. An element consists of a value paired with a non-empty label.
2. In general, a record may contain any number of element instances bearing the same label.
3. Element order is preserved.

In addition to these element rules, an ERC is considered complete only if all four elements "who", "what", "when", and "where" are present with no missing values; these four h's each have the coded synonyms h1, h2, h3, and h4, respectively. If a best effort to supply a value fails, in its place must be given a standardized value (below) indicating the reason for the missing value.

As mentioned, the Kernel is just a vocabulary and it is the ERC that imposes assumptions about required elements. The four h's may be supplied with implicit labels by using the abbreviated-form ERC. In this case, element ordering must be strictly observed, as in

```
erc:  Lederberg, Joshua
      | Studies of Human Families for Genetic Linkage | 1974
      http://profiles.nlm.nih.gov/BB/AA/TT/tt.pdf
note:   This is an arbitrary note inside a
        small descriptive record.
```

A record that does not have all four h's is considered a "stub ERC". Stubs may be especially useful for holding records that are under construction or are subject to an automated completion process.

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 (e.g., on demand) local records to an ERC container 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.

---

7. Kernel Label Structure

The rest of this document is concerned with Kernel metadata independent of the ERC. Nonetheless, examples will continue to be given using the ANVL/ERC format.

Kernel element labels are strings beginning with a letter that may contain any combination of letters, numbers, hyphens, and underscores ("_"). Element labels are therefore fairly consistent with the rules for **[XML]**. An inconsistency is that Kernel labels may be entered with spaces; in this case all sequences of spaces are considered equivalent to a single space, and that space in turn is then considered (for matching and for export to XML) to be equivalent to an underscore. Any initial and final spaces are stripped away before processing a label.

For comparison purposes, element labels are also considered case-insensitive; in other words, labels may be entered and displayed with case differences, but there is no possibility of conflict behind the scenes when spaces and upper case are normalized to underscore and lower case. For example, these rules prevent any future version of the Kernel from ever having these as two distinct elements,

```
marc_856
MARC 856
```

For display purposes, element labels are considered case-sensitive; in other words, upper- and lower-case distinctions should be preserved upon display.

An element label may also be accompanied by its coded synonym. In ANVL the synonym follows the label and is enclosed in parentheses (whereis in XML, for example, the synonym might be an XML attribute). In fact, if the official coded synonym is present, the label itself may be represented in any UTF-8 **[RFC3629]** form (e.g., in a local translation) that is convenient for the record's local audience, as in,

```
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(h501): (en)  For your Own Good:  Hidden Cruelty
             in Child-Rearing and the Roots of Violence
```

In this example, the labels are intended for local audiences and the coded synonyms allow for unambiguous interpretation by software that can display labels translated for other audiences.

---

8. Kernel Sort-Friendly Values

To keep records easy to sort and survey, it helps if element values are somewhat comparable. To this end the Kernel strongly encourages values that are "sort-friendly". In this way, applications have a reasonable chance of successfully presenting a set of given records sorted according to specific element values, such as date or author name, with only general-purpose software that need not make special assumptions about the structure and form of the values. It is therefore standard to assume that the creator of Kernel metadata has made a best effort to enter dates, titles, and names in a sort-friendly manner. For example, these values are easy for non-special-purpose sorting software to handle,

```
who:  Khan, Hashim
when: 19580924
```

while these values are not sort-friendly,

```
who:  Hashim Khan
when: Sep 24, 1958
```

8.1. Initial Comma to Recover Natural Word Order

Sometimes the desire to create sort-friendly values conflicts with natural word order, such as with Western-style personal names. To mitigate this concern, a value may optionally begin with a "," (comma) to indicate how to recover natural word order; the way it works is, if other commas are present in the value, they mark inversion points that software (or the human eye) can use to re-order words in the value. For example,

```
who:,  van Gogh, Vincent
```

```
who:,  Howell, III, PhD, 1922-1987, Thurston
who:,  Acme Rocket Factory, Inc., The
who:,  Mao Tse Tung
```

Natural word order can be restored by taking the last non-empty part of the value set off by an internal comma and placing it at the beginning. At times a secondary sort point (such as a name given within a family) would be useful but is blocked because that position in the value is taken by an insignificant word (such as "Dr" or "Mr"). The remedy is to bracket the insignificant word with commas and place it at the end where naive sorting software would then treat it with minimum significance. For example, in these cases,

```
who:,  McCartney, Pat, Ms,
who:,  McCartney, Paul, Sir,
who:,  McCartney, Petra, Dr,
what:, Health and Human Services, United States Government
          Department of, The,
```

natural word order is restored by first pulling off any final value part bracketed by commas, applying the previous rule, and then adding back that final part to the beginning. The values from the above two sets of examples have the following natural word orders.

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

This feature is typically used to express Western-style personal names in family-name-given-name order. As the last line above shows, it can also be used wherever natural word order might not work with naive sorting software, such as when data contains titles or corporate names.

While Kernel metadata creators should make a best-effort to produce values that are sort-friendly when compared with the same element in other records, the consequences of deviating from this need not be serious. For instance, it is usually more useful to supply a value for an element than to suppress it merely because it won't necessarily sort well when records appear in groups.

---

9. Kernel Value Structure

With sort-friendliness as a secondary criterion, in general Kernel values consist of free text. Exceptions are triggered by structuring markers that may occur either anywhere inside a value or only at the beginning of a value.

Markers that may occur anywhere in a value:

> ";" for *multiple values* and
> "|" for *subvalues*

Markers that may occur only at the beginning of a value:

> "(: ...)" for special *value indicators* or
> one of the characters ";", "|", or "," explained later.

These structuring markers are explained next.

9.1. Multiple Values and Subvalues

The semi-colon (";") is used to separate multiple "peer" values that could equivalently be represented as multiple elements with the label repeated for each separate value; in programmer terms, the ";" is a kind of *array* element separator. For example,

```
who:  Smith, J; Wong, D; Khan, H
```

is a shorter way of representing

```
who:  Smith, J
who:  Wong, D
who:  Khan, H
```

The solidus ("|") is used to separate component subvalues with different types of "non-peer" contribution to the overall value; this supports an element that has sub-structure. For example,

```
in:  EEG Clin Neurophysiol | v103, i6, p661-678 | 19971200
```

If used together, ";" holds its neighbors more tightly (has higher grouping precedence) than "|". For example, in this "erc" element

```
erc:  Smith, J; Wong, D; Khan, H
     | Cocktail Napkin Drawing #2 | 1969
     | (:unav) destroyed during spill of 19690401
```

there are four sub-elements, the first of which has three repeated values.

9.2. Kernel Initial Value Conventions

Kernel values usually start with free text, but exceptions are made when the first character of a value begins with one of the single action characters ";", "|", or ",". When one of the single characters is recognized at the start of a value, the appropriate action is taken, the character is effectively removed, and processing continues on the remainder until a character that is not one of these three is seen. For example, once a SPACE character or a "(: ...)" construct (a special value indicator) has been recognized, no further initial single character processing occurs.

When a value or subvalue starts with ";", it "quotes" any internal occurrences of ";", in other words, it turns off the special ability of ";" to divide a value or subvalue into multiple values. When a value starts with "|", it "quotes" any internal occurrences of "|", in other words, it turns off the special ability of "|" to divide a value into subvalues.

When a value or subvalue starts with ",", it indicates a way to recover natural word order, as explained previously.

9.3. Special Kernel Standardized Value Codes

A value starting with "(: ...)" indicates a standardized (controlled) value code, usually short and precise, that is designed to be readable by software. Such a value code often forms only part of the value. More than one value code may appear at the beginning of a value.

Special value codes serve different purposes. A code can indicate a single specific value, with the remaining value text offering a human-readable equivalent; for example,

```
   who:  (:unkn) anonymous
```

tells software that the element value is officially unknown and the other text tells the same thing to a human reader of English that may be expecting the name of an author. A code can also indicate that the value is at a location given by the remaining text (which should be an actionable identifier such as a URL) and is not otherwise present; for example,

```
   who:     Wong, D
   who:     (:at) http://example.org/abc/def/ghi.txt
   rights:  (:at) http://example.com/rights/123.html
```

could be used to indicate a first author, sixty-five co-authors listed in a separate file, and a copyright statement posted on a corporate website.

Some special value codes are summarized here. All but the last four indicate different kinds of "missing value":

(:unac) temporarily inaccessible
(:unal) unallowed, suppressed intentionally
(:unap) not applicable, makes no sense
(:unas) value unassigned (e.g., Untitled)
(:unav) value unavailable, possibly unknown
(:unkn) known to be unknown (e.g., Anonymous, Inconnue)
(:none) never had a value, never will
(:null) explicitly and meaningfully empty
(:tba) to be assigned or announced later
(:etal) too numerous to list (et alia).
(:at) the real value is at the given URL or identifier.

9.4. Kernel Date Values

A commonly recurring value type is a date, which may be followed by a time. The **[TEMPER]** format is preferred to the **[W3CDTF]** format, which has limitations in expressing ranges, lists, approximate, and BC dates. Kernel dates may take one of the following forms:

```
1999               (four digit year)
20001229           (year, month, day)
20001229235955     (year, month, day, hour, minute, second)
```

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

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

Approximate and BCE dates can also be expressed, as in,

```
1850~                       (around the year 1850)
BCE1212                     (death of Rameses the Great)
BCE0551                     (birth of Confucius)
```

Note that BCE dates inherently sort in reverse order. But because "BCE" appears first in the TEMPER value, naive sorting software first places all BCE dates together as a group, after which the simple intervention of reversing the order of the group achieves correct chronological order.

9.5. Element Value Encoding

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

The value encoding mechanism also uses `%', but instead of taking two following hexadecimal digits, it takes two alphabetic characters that cannot be mistaken for hex digits or one non-alphanumeric character. It is designed not to be confused with normal web-style %-encoding. In particular it can be decoded without risking unintended decoding of normal %-encoded data (which would introduce errors). Here are the extended Kernel encoding extensions, the middle column giving the equivalent and usual hexadecimal encoding.

```
 Code   Hex     Purpose
 ----   ---    ------------------------------------------------
 %sp    %20    decodes to space
 %ex    %21    decodes to !
 %dq    %22    decodes to "
 %ns    %23    decodes to #
 %do    %24    decodes to $
 %pe    %25    decodes to %
 %am    %26    decodes to &
 %sq    %27    decodes to '
 %op    %28    decodes to (
 %cp    %29    decodes to )
 %as    %2a    decodes to *
 %pl    %2b    decodes to +
 %co    %2c    decodes to ,
 %sl    %2f    decodes to /
 %cn    %3a    decodes to :
 %sc    %3b    decodes to ;
 %lt    %3c    decodes to <
 %eq    %3d    decodes to =
 %gt    %3e    decodes to >
 %qu    %3f    decodes to ?
 %at    %40    decodes to @
 %ox    %5b    decodes to [
 %ls    %5c    decodes to \
 %cx    %5d    decodes to ]
 %vb    %7c    decodes to |
 %nu    %00    decodes to null
 %%     %25    decodes to %
 %_     n/a    a non-character used as a syntax shim
 %{     n/a    a non-character that begins an expansion block
 %}     n/a    a non-character that ends an expansion block
```

One particularly useful construct in an element values is the pair of special encoding markers ("%{" and "%}") that indicates a "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 readable way. For example, the value in

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

is decoded into an equivalent element, but with a correct and intact URL:

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

---

10. Kernel Changes New in this Specification (Sept 2007)

1. editorial changes based on feedback from DC 2007 and discussion in the Kernel Application Profile task group
2. coined a URI base (currently not actionable) as a unique reference for each vocabulary term, partly in order to prepare a DCMI Kernel Application Profile

3. added reference to ARK persistent identifier scheme, which uses Kernel/ERC
4. addition of "(:" and ")" in relevant vocabulary entries
5. eliminated unneeded initial character escaping ambiguity; to prevent initial single-character processing of ",", ";", and "|", it is sufficient to begin a subvalue with a SPACE

---

11. Vocabulary of Elements and Values

This vocabulary includes a mixture of Kernel elements, values, and concepts. In the definitions below, the term "resource" is synonymous with "object". Each vocabulary element label has a short, coded synonym that consists of the letter 'h' followed by a number, such as h1, h2, h3, etc. Each vocabulary element also has a long, globally unique identifier that is a URI composed of http://n2t.info/ark:/99152/ followed by the short synonym; for example,

```
about-when(h13)  -->  http://n2t.info/ark:/99152/h13
```

At the price of some redundancy, it also includes the basic 15 Dublin Core (DC) element definitions because (a) DC elements can be used without namespace qualification in ERC records and (b) the Kernel assigns them coded synonyms (h501-h515).

about-erc (h10):
: A composite element, structured according to the four h's, that describes the content of the object. Without a value, it is a label for visually setting off a region in a record.

about-what (h12):
: A topic of the resource. DC Mapping: Subject

about-when (h13):
: A temporal topic of the resource. DC Mapping: Coverage (temporal)

about-where (h14):
: A spatial topic of the resource. DC Mapping: Coverage (spatial)

about-who (h11):
: A name of a personage that is a topic of resource.

about-how (h15):
: An account of the resource. DC Mapping: Description

contributor (h506):
: An entity responsible for making contributions to the resource. Examples of a Contributor include a person, an organization, or a service. Typically, the name of a Contributor should be used to indicate the entity.

coverage (h514):
: The spatial or temporal topic of the resource, the spatial applicability of the resource, or the jurisdiction under which the resource is relevant. Spatial topic and spatial applicability may be a named place or a location specified by its geographic coordinates. Temporal topic may be a named period, date, or date range. A jurisdiction may be a named administrative entity or a geographic place to which the resource applies. Recommended best practice is to use a controlled vocabulary such as the Thesaurus of Geographic Names [TGN]. Where appropriate, named places or time periods can be used in preference to numeric identifiers such as sets of coordinates or date ranges.

creator (h502):
: An entity primarily responsible for making the resource. Examples of a Creator include a person, an organization, or a service. Typically, the name of a Creator should be used to indicate the entity.

date (h507):
: A point or period of time associated with an event in the lifecycle of the resource. Date may be used to express temporal information at any level of granularity. Recommended best practice is to use an encoding scheme, such as the W3CDTF profile of ISO 8601 [W3CDTF].

description (h504):
: An account of the resource. Description may include but is not limited to: an abstract, a table of contents, a graphical representation, or a free-text account of the resource.

ERC
: Electronic Resource Citation, an object description that uses, at a minimum, the fundamental Kernel elements, who, what, when, and where addressing the expression of the object.

erc (h0):
: A composite element, structured according to the four h's, that describes the expression of the resource. Without a value, it is a label declaring a record to be an ERC, a complete instance of which requires non-missing values for each of the four h's.

(:etal)
: A null element term explaining that the value is a stand-in for other values too numerous to list (et alia).

format (h509):
: The file format, physical medium, or dimensions of the resource. Examples of dimensions include size and duration. Recommended best practice is to use a controlled vocabulary such as the list of Internet Media Types [MIME].

four h's
: The four fundamental Kernel elements — who, what, when, where — commonly used to structure composite Kernel elements. To say "structured according to the four h's" indicates a sub-element sequence suggesting this particular sequence; this serves as an important memory aid with abbreviated form elements in which explicit labels are absent. The literal form of these labels, by themselves, address the story of the expression of an object, and in that form they are required of every complete ERC. Future versions of the Kernel may extend the sequencing of four h's with non-required elements "how" and "why".

identifier (h510):
: An unambiguous reference to the resource within a given context. Recommended best practice is to identify the resource by means of a string conforming to a formal identification system.

in (h602):
: (under construction) Reserved for a composite element referencing a serial publication in which the described object appears. This element is structured in a manner loosely reminiscent of the four h's, indicating serial name, volume/issue/page, date, and issue URL. DC Mapping: Relation

how (h5):
: (under construction) Reserved for a coded value indicating how the object was expressed.

language (h512):

A language of the resource. Recommended best practice is to use a controlled vocabulary such as RFC 4646 [RFC4646].

metadata
: Structured data, generally descriptive of or associated with a given object or resource. Structured data at a minimum has evident start and end points and may have evident labels.

meta-erc (h30):
: A composite element, structured according to the four h's, that describes the expression of this (the containing) record. Without a value, it is a label for visually setting off a region in a record.

meta-what (h32):
: A short form of the identifier for the record.

meta-when (h33):
: The last modification or review date of the record.

meta-where (h34):
: A location of the fullest form of the record.

meta-who (h31):
: A person or party responsible for the record.

(:none)
: A null element term explaining that the element never had a value and never will. This is a stronger form of :unas.

note (h601):
: A free text note about the record.

(:null)
: A null element term explaining that the value is explicitly empty, where an empty value has a well-defined meaning in contexts (not necessarily evident) in which the element is used.

object
: Anything to which metadata may be applied. Synonym: "resource"

publisher (h505):
: An entity responsible for making the resource available. Examples of a Publisher include a person, an organization, or a service. Typically, the name of a Publisher should be used to indicate the entity.

resource
: Anything to which metadata may be applied. Synonym: "object"

relation (h513):
: A related resource. Recommended best practice is to identify the related resource by means of a string conforming to a formal identification system.

rights (h515):
: Information about rights held in and over the resource. Typically, rights information includes a statement about various property rights associated with the resource, including intellectual property rights.

source (h511):
: A related resource from which the described resource is derived. The described resource may be derived from the related resource in whole or in part. Recommended best practice is to identify the related resource by means of a string conforming to a formal identification system.

subject (h503):
: The topic of the resource. Typically, the subject will be represented using keywords, key phrases, or classification codes. Recommended best practice is to use a controlled vocabulary. To describe the spatial or temporal topic of the resource, use the Coverage element.

support-erc (h20):
: A composite element, structured according to the four h's, that describes the support commitment a provider makes to the object. Without a value, it is a label for visually setting off a region in a record.

support-what (h22):
: A short form of the commitment made to the object.

support-when (h23):
: The last modification or review date of the commitment made to the object.

support-where (h24):
: A location of the fullest form of the commitment made to the object.

support-who (h21):
: A person or party responsible for the object, such as the provider of preservation or access services.

stub ERC
: An incomplete ERC record. To be incomplete it is sufficient for one or more of the four h's (the elements who, what, when, and where) to be missing or to have a missing value.

(:tba)
: A null element term explaining that the value is to be assigned or announced later.

title (h501):
: A name given to the resource.

type (h508):
: The nature or genre of the resource. Recommended best practice is to use a controlled vocabulary such as the DCMI Type Vocabulary [DCTYPE]. To describe the file format, physical medium, or dimensions of the resource, use the Format element.

(:unac)
: A null element term explaining that the value is temporarily inaccessible. This might be due, for example, to a system outage.

(:unal)
: A null element term explaining that the value is unallowed or suppressed intentionally.

(:unap)
: A null element term explaining that no value is applicable or makes no sense.

(:unas)
: A null element term explaining that a value was never assigned. An untitled painting is an example.

(:unav)
: A null element term explaining that the value is unavailable for some reason. Compared to :unkn, this term conveys no particular confidence about the non-existence of the value. It may originate in collections that have not yet conducted a thorough investigation or it may arise in intermediate systems that repackage received records having missing elements.

(:unkn)
: A null element term explaining that the value is unknown. Compared to :unav, this term conveys greater confidence and authority that an appropriate value is unknown to anyone for the object described. An example is an expert assessment of "anonymous" concerning authorship.

what (h2):
: A human-oriented name given to the resource, or what this expression of the resource was called. Compared to the "where"

element, which is also a kind of name, the "what" element tends to be more suitable for human consumption. DC Mapping: Title

when (h3):
: A point or period of time associated with an event in the lifecycle of the resource, often when it was expressed, created or made available. DC Mapping: Date

where (h4):
: An access-oriented name given to the resource, or where this resource was expressed. is to identify the resource by means of a string or number conforming to a formal identification system. Compared to the "what" element, which is also a kind of name, the "where" element tends to be more suitable for automated access. DC Mapping: Identifier

who (h1):
: An entity responsible for expressing the object, such as creating it or making it available. Examples of "who" include a person, an organization, or a service. DC Mapping: Creator, but if no Creator use Publisher, and if no Publisher, use Contributor

---

12. References

[AACR2] American Library Association, "**Anglo-American Cataloguing Rules**," 2007 (**HTML**).

[ANVL] Kunze, J. and Kahle, B., "**A Name-Value Language**," February 2005 (**PDF**).

[ARK] Kunze, J. and R. Rodgers, "**The ARK Persistent Identifier Scheme**," July 2007 (**PDF**).

[DCMI] Dublin Core Metadata Initiative, "**DCMI Metadata Terms**" (**HTML**).

[MARC] Library of Congress, "**Machine Readable Cataloguing**," 2007 (**HTML**).

[MODS] Library of Congress, "**Metadata Object Description Schema**," June 2006 (**HTML**).

[PREMIS] OCLC and RLG, "**PREMIS Data Dictionary, version 1.0**," 2005 (**PDF**).

[RDF] W3C, "**Resource Description Framework**" (**HTML**).

[TEMPER] Blair, C. and J. Kunze, "**Temporal Enumerated Ranges**," August 2007 (**PDF**).

[W3CDTF] "**Date and Time Formats (W3C profile of ISO8601)**" (**HTML**).

[XML] W3C, "**Extensible Markup Language (XML) 1.0 (Fourth Edition)**," August 2006 (**HTML**).

[RFC5013] Kunze, J. and T. Baker, "**The Dublin Core Metadata Element Set**," RFC 5013, August 2007.

[RFC2822] Resnick, P., "**Internet Message Format**," RFC 2822, April 2001.

[RFC3629] Yergeau, F., "**UTF-8, a transformation format of ISO 10646**," STD 63, RFC 3629, November 2003.

[RFC3986] **Berners-Lee, T.**, **Fielding, R.**, and **L. Masinter**, "**Uniform Resource Identifier (URI): Generic Syntax**," STD 66, RFC 3986, January 2005 (**TXT**, **HTML**, **XML**).

Authors' Addresses

John A. Kunze
California Digital Library
415 20th St, 4th Floor
Oakland, CA 94612
US
Fax: +1 510-893-5212
Email: jak@ucop.edu

Adrian Turner
California Digital Library
415 20th St, 4th Floor
Oakland, CA 94612
US
Fax: +1 503-234-3581
Email: adrian.turner@ucop.edu