DCMI Kernel Metadata Community	J. Kunze G. Janée A. Turner California Digital Library
	November 25, 2014

TOC

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

TOC

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]**. With origins in **[EPerm]**, the 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 the "URI" **[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 an act of *expressing* the object. This sequence applied to various Kernel "stories" is a recurring theme (cf. **[ISO19773]** event

descriptions). In anticipation of future extensions to "how", "why", and "huh", 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 isolated 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

TOC

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. Two applications that use Kernel metadata are **[ARK]** identifiers and Namaste (NAMe-AS-TExt) tags **[Namaste]**.

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.

---

3. Kernel Stories

TOC

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 "telling" — and in so doing form something similar to a traditional citation:

> *who* "told" it (from DC Creator, Contributor, and Publisher),
> *what* the "telling" was called (from DC Title),
> *when* it was "told" (from DC Date), and
> *where* the "telling" should be found (from DC Identifier).

(A word such as "expression" might work better than "telling" for this metaphor, but it conflicts with existing bibliographic jargon.)

One descriptive record may contain stories of different forms of the same object, for example, its digital and physical forms. Depending on the object type — article, photograph, dance, fossil — an object's "telling" 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, the hosting organization's support for the object, and information about the depositor. More about these story types is given after first describing the story that anchors a descriptive record.

3.1. The Anchoring Story

TOC

Among all the tales that an object description may tell, the *anchoring story* is the one, usually decided upon deposit of an object into a collection, that was deemed the most suitable basic object referent for the purposes of orderly collection management. There are different kinds of object managers, from subject specialists who create objects, to digital library experts who acquire them, to collection and system staff who monitor them. Who among these will decide what metadata comprises the anchoring story will often be determined by local collection workflow practices.

Kernel metadata is primarily concerned with supporting orderly collection curation as a partnership between object depositors and repository managers. This does not assume that the depositor or manager, who between them will do the bulk of object management, has any training in bibliographic description or in collection subject areas. When an object throws a system exception or appears out of place in a depositor-oriented asset report, Kernel metadata is designed to permit either party to form a clear enough mental picture to conduct a telephone conversation about it.

The anchoring story is a "lay" description designed to support this depositor-manager partnership. If there is no other metadata, the anchoring story should suffice. Even if

there is more and richer metadata, the anchoring story should still be created for those repositories that lack specialized staff resources to re-catalog or to develop "cross-walks" that will transform the richer metadata into an anchoring story. This anchoring story, expressed via the four h's needs to work for depositor reports, system logs, object "tombstones" (what is left when the object is withdrawn), and a record of an object's birth (arrival) in the respository. The anchoring story can be thought of as "birthstone" metadata.

The depositor has great latitude in choosing its anchoring story, but it should appear 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, with the "where" element adjusted to the identifier of the object in the current repository (after deposit).

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 "where" of the "telling" should, if possible, be a persistent, machine-actionable identifier to a digital experience of the object. This provides a lay description with the convenience of online access, while avoiding the tedious and uninteresting details about who digitized the object, when it was scanned, etc). The state of an object's having only a physical instance is tranisient; the most interesting physical objects are often only a couple of funding cycles away from the creation of some sort of digital experience of the object. The complete and pure stories of both the derivative and original objects can be told, if necessary and affordable, elsewhere in the record.

An anchoring story need not be the central descriptive goal of an ERC (or any other) record. For example, a museum depositor 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 depositor 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 depositor 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 form, it makes sense for the provider to choose a suitable online edition of the play to tell the anchoring story for the ERC.

3.2. Story Summary

TOC

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 persistent, system-oriented identifier
(required)
 how:   (under construction) a formal type designator
```

Another story is that of an object's content.

```
 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
```

Another story is that of the origin of the metadata record itself.

```
 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 persistent identifier of the fullest
form of the record
```

Another story is that of a support commitment made to an object.

```
 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 persistent identifier of the
fullest form of the commitment
```

Another story is that of the depositor.

```
 depositor-who:   a person or party responsible for
the deposit
 depositor-what:  role of depositor in depositing
organization
 depositor-when:  dates of depositor's tenure in that
role
 depositor-where: a unique machine-readable depositor
identifier
```

4. Kernel Summary and Dublin Core Crosswalk

TOC

Each Kernel element label has a globally unique identifier of the form http://n2t.net/ark:/99152/h*N*, where *N* corresponds to the terms given in the vocabulary section later in this document. For example, the Kernel label "who" has the linked data URI, http://n2t.net/ark:/99152/h1.

These identifiers should be used to reference Kernel elements in all linked data applications **[SEMWEB]**. The namespace defined by URIs starting with http://n2t.net/ark:/99152/ is permanently reserved for metadata terms by the California Digital Library (maintenance agency for the ARK identifier scheme).

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 APPROXIMATION		KERNEL LABEL	SYN	DUBLIN CORE
erc: Creator/Contributor/Publisher	*	who	h1	
The story of an object's "telling". (permanent)	*	what	h2	Title
	*	when	h3	Date
	*	where	h4	Identifier
		how	h5	(reserved Type restriction**)
about-erc: (personage)		about-who	h11	Subject
The story of an object's (temporal) content. (spatial)		about-what	h12	Subject
		about-when	h13	Coverage
		about-where	h14	Coverage
		about-how	h15	Description

```
*  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]**).

4.1. Kernel and Dublin Core Interoperability

TOC

Kernel metadata currently maintains a basic level of interoperability with Dublin Core metadata. In formal terms, it conforms to "Level 1" interoperability **[DCMI-IL]**. Higher levels of interoperability, such as the "Level 2" mapping of Kernel metadata to RDF, are being sought and may be established in subsequent specifications.

---

5. The Kernel and the ERC

TOC

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 (unlike 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

TOC

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 is possible to represent an ERC in many different encodings (e.g., XML with the schema in Appendix A), 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 (two

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 (a programmer would call this a *comment* line).
5. The ordering of lines is significant; e.g., the ordering should be preserved upon record display or transformation.

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

TOC

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 ("_"). A period (".") is reserved to separate a namespace designation from a label and must otherwise be encoded as "%pd". Element labels are therefore fairly consistent with rules for **[XML]** names.

One inconsistency with XML 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 (whereas 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

TOC

Metadata standards can in principle guide the creation of records that result in sensible orderings when sorted by such things as title, date, and author, however, some standards are not prescriptive in this regard (e.g., Dublin Core **[RFC5013]**). Moreover, when received metadata originates from a wide variety of sources and domains, as in "semantic web" applications, incompatible creation practices make it hard to produce completely understandable orderings.

For Kernel metadata, a set of values (e.g., author names found across a set of records) is considered *sort-friendly* if a simple lexical sort results in an ordering that makes sense to users. While this definition leaves room for subjectivity and variability in creation practices, it demands consistency within each local practice. More importantly, without the benefit of domain or source knowledge, any receiver with the simplest tools is likely to be able to sort these values sensibly, even if they originate from otherwise very different practices. As an example, simple lexical sorting by any one of the element forms,

```
who:  Khan, Hashim
what: Importance of Being Earnest, The
when: 19580924
```

would create alphabetical orderings by either "most significant word" or chronological ordering. This readily understandable result suggests that the set of values

represented by each of these forms is sort-friendly. By contrast, simple sorting by any one of these forms,

```
who:  Hashim Khan
what: The Importance of Being Earnest
when: Sep 24, 1958
```

would create orderings that will be alphabetic by given name, first title word (significant or not), or month name. For a few special applications, this may be the intended effect, but in most cases it will be a sign of values that are not sort-friendly. There is a tension in Western languages, addressed in the next section, between sort-friendly values and natural word order.

Creators of Kernel metadata are assumed to have made a best effort to include dates, titles, names, and other values in a sort-friendly manner. This does not solve the difficult general problem of creating fully sortable, cross-domain records, but it is a practical first step. Sort-friendly values can be equally useful for Kernel and non-Kernel metadata.

8.1. Commas to Recover Natural Word Order

TOC

Sometimes the desire to create sort-friendly values conflicts with natural word order, as with Western-style personal names and grammatical articles for which less significant words precede (stealing sort precedence from) more significant words. To mitigate this conflict, a value may optionally end with a "," (comma) that indicates how to recover natural word order. It works roughly as follows: if other non-final 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:  Hu Jintao,
```

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. Note that if there are no commas at all or only one comma is present, no inversion point is indicated.

If two inversion points are desired, end the value with two commas in a row. This can help when there are three grades of word significance, as with Western honorifics. The two inversion points are positioned so that the second to last part serves as a secondary sort key and the last part as a tertiary sort key. To recover natural word order, the second to last non-empty part of the value bracketed by commas is placed at the beginning, preceded by the very last non-empty part. 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 the final non-empty part bracketed by commas, applying the previous rule (moving the now-final non-empty part to the beginning), and then adding back that formerly-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.
Hu Jintao
Ms Pat McCartney
Sir Paul McCartney
Dr Petra McCartney
The United States Government Department of Health and
Human Services
```

As mentioned this feature is can be used to express Western-style personal names in family-name-given-name order. The last line above shows that it can also be used

wherever natural word order might produce unexpected results 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 failing to do so need not halt metadata production. It is more useful to supply a value for an element than to suppress it because of uncertainty about whether it will sort well.

---

9. Kernel Value Structure

TOC

With sort-friendliness as a desirable, 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 *repeated 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. Alternate Values, Repeated Values, and Subvalues

TOC

The semi-colon (";") is used to separate repeated "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. In mapping between these single- and multi-line forms, ordering of element values should be preserved. 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.

At an even higher level of precedence than ";" is "(=)", which is used to separate alternate versions of one particular value. For example, in

```
who:  Bibliothèque nationale de France (=) National
Library
         of France (=) BNF; Library of Congress (=)
LC
```

the first of two repeated values contains an institutional name in French, then in English, and then as an acronym. In the second of the two repeated values are two alternate values.

9.2. Kernel Initial Value Conventions

TOC

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 repeated 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. Similarly, when a value or subvalue starts with ",", it turns off the special ability of "," at the end to indicate word order inversion points, as explained previously.

9.3. Special Kernel Standardized Value Codes

TOC

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

TOC

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

TOC

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 **[RFC3986]** 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 ,
 %pd    %2e    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&qu
ery=foo+bar+zaf
```

---

10. Vocabulary of Elements and Values

TOC

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.net/ark:/99152/ followed by the short synonym; for example,

```
about-when(h13)   -->  http://n2t.net/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 the 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].

depositor-erc (h40):
: A composite element, structured according to the four h's, that describes the depositor of the object.

depositor-who (h41):
: The name of a person or party responsible for the deposit.

depositor-what (h42):
: The role of the depositor in the depositing organization.

depositor-when (h43):
: The dates of depositor's tenure in the role of depositor within the depositing organization.

depositor-where (h44):
: A unique machine-readable identifier for the depositor.

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 "telling" of the object.

erc (h9):
: A composite element, structured according to the four h's, that describes the "telling" 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 **[RFC2046]**.

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 "telling" 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", "why", and "huh".

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.

huh (h7):
: (under construction) Reserved to indicate the character set encoding and language of the metadata record.

language (h512):
: A language of the resource. Recommended best practice is to use a controlled vocabulary such as **[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 "telling" 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 persistent identifier 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 persistent identifier 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 "telling" 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

why (h6):
: (under construction) Reserved for required legal language that must appear in a metadata record, including copyright and disclaimer statements.

11. References

TOC

[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] DCMI Usage Board, "**DCMI Metadata Terms**" (**HTML**).

[DCMI-IL] DCMI Usage Board, "**Interoperability Levels for Dublin Core Metadata**," May 2009 (**HTML**).

[DCTYPE] DCMI Usage Board, "**DCMI Type Vocabulary**" (**HTML**).

[EPerm] Kunze, J., "**A Metadata Kernel for Electronic Permanence**," October 2001 (**HTML**).

[ISO19773] ISO/IEC, "**ISO/IEC 19773:2011 Information technology - Metadata Registries (MDR) modules**," 2011 (**HTML**).

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

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

[Namaste] Kunze, J., "**Directory Description with Namaste Tags**," April 2009 (**HTML**).

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

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

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

[RFC2046] **Freed, N.** and **N. Borenstein**, "**Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types**," RFC 2046, November 1996 (**TXT**).

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

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

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

[RFC4646] Phillips, A. and M. Davis, "**Tags for Identifying Languages**," RFC 4646, September 2006 (**TXT**).

[SEMWEB] Shadbolt, N., Hall, W., and T. Berners-Lee, "**The Semantic Web Revisited**," May 2006 (**HTML**).

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

[TGN] Getty, J., "**Thesaurus of Geographic Names**" (**HTML**).

[W3CDTF] Wolf, M. and C. Wicksteed, "**Date and Time Formats (W3C profile of ISO8601)**" (**HTML**).

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

Appendix A. ERC XML Schema V1.0

TOC

This section contains an XML schema for the ERC.

```
<?xml version="1.0" encoding="UTF-8"?>

<schema xmlns="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://dublincore.org/dcx/kernel/"
  elementFormDefault="qualified">

<element name="erc">
  <complexType>
    <sequence>
      <element name="who" type="string"
maxOccurs="unbounded"/>
      <element name="what" type="string"
maxOccurs="unbounded"/>
      <element name="when" type="string"
maxOccurs="unbounded"/>
      <element name="where" type="string"
          minOccurs="0" maxOccurs="unbounded"/>
      <element name="how" type="string"
          minOccurs="0" maxOccurs="unbounded"/>
      <element name="about-who" type="string"
          minOccurs="0" maxOccurs="unbounded"/>
      <element name="about-what" type="string"
          minOccurs="0" maxOccurs="unbounded"/>
      <element name="about-when" type="string"
          minOccurs="0" maxOccurs="unbounded"/>
      <element name="about-where" type="string"
          minOccurs="0" maxOccurs="unbounded"/>
      <element name="about-how" type="string"
          minOccurs="0" maxOccurs="unbounded"/>
      <element name="meta-who" type="string"
          minOccurs="0" maxOccurs="unbounded"/>
      <element name="meta-what" type="string"
          minOccurs="0" maxOccurs="unbounded"/>
      <element name="meta-when" type="string"
          minOccurs="0" maxOccurs="unbounded"/>
      <element name="meta-where" type="string"
          minOccurs="0" maxOccurs="unbounded"/>
      <element name="support-who" type="string"
          minOccurs="0" maxOccurs="unbounded"/>
      <element name="support-what" type="string"
          minOccurs="0" maxOccurs="unbounded"/>
      <element name="support-when" type="string"
```

```
          minOccurs="0" maxOccurs="unbounded"/>
      <element name="support-where" type="string"
          minOccurs="0" maxOccurs="unbounded"/>
      <element name="depositor-who" type="string"
          minOccurs="0" maxOccurs="unbounded"/>
      <element name="depositor-what" type="string"
          minOccurs="0" maxOccurs="unbounded"/>
      <element name="depositor-when" type="string"
          minOccurs="0" maxOccurs="unbounded"/>
      <element name="depositor-where" type="string"
          minOccurs="0" maxOccurs="unbounded"/>
    </sequence>
  </complexType>
</element>

</schema>
```

Authors' Addresses

TOC

John A. Kunze
California Digital Library
415 20th St, 4th Floor
Oakland, CA 94612
US
Email: jak@ucop.edu

Greg Janée
California Digital Library
415 20th St, 4th Floor
Oakland, CA 94612
US
Email: greg.janee@ucop.edu

Adrian Turner
California Digital Library
415 20th St, 4th Floor
Oakland, CA 94612
US
Email: adrian.turner@ucop.edu