Topic: Supplementary meeting packet
Identifier: http://www.bi.fhg.de/People/Thomas.Baker/agenda/topic-supplementary/
See also: http://www.bi.fhg.de/People/Thomas.Baker/agenda/
Created: 2005-05-13
Modified: 2005-05-13 11:12, Friday
Maintainer: Tom Baker

-- DCMI Namespace Policy
http://dublincore.org/documents/dcmi-namespace/

-- DCMI Abstract Model (DCMI Recommendation)
http://dublincore.org/documents/abstract-model/

-- DCMI Policy on Naming Terms
http://dublincore.org/documents/naming-policy/

-- Guidelines for Assigning Identifiers to Metadata Terms
http://www.ukoln.ac.uk/metadata/dcmi/term-identifier-guidelines/

-- Dublin Core Application Profile Guidelines
http://dublincore.org/usage/meetings/2004/03/cwa14855-20040210.pdf

Topic: Agenda for Usage Board meeting, Washington DC, 19-20 May 2005
Identifier: http://www.bi.fhg.de/People/Thomas.Baker/agenda/
Created: 2005-05-13
Modified: 2005-05-13 11:12, Friday
Maintainer: Tom Baker

2005-05-19: 10:00-12:00 (2 hours)

01. Accessibility [Tom]
http://www.bi.fhg.de/People/Thomas.Baker/agenda/topic-accessibility

2005-05-19: 13:00-17:30 (4 hours)

02. DCMI Extension Namespaces and review of application profiles [Tom]
http://www.bi.fhg.de/People/Thomas.Baker/agenda/topic-profiles

03. Review of Collection Description Application Profile [all]
http://www.bi.fhg.de/People/Thomas.Baker/agenda/topic-profiles-collection

04. Review of Library Application Profile [Tom]
http://www.bi.fhg.de/People/Thomas.Baker/agenda/topic-profiles-libraries

2005-05-20: 9:00-12:00 (2+ hours)

05. MARC relator terms - 1.5 hours [Rebecca]
http://www.bi.fhg.de/People/Thomas.Baker/agenda/topic-relators

06. dc:language - 30 min [Rebecca]
http://www.bi.fhg.de/People/Thomas.Baker/agenda/topic-language-comment

07. Type vocabulary - 1 hr [Stuart]
http://www.bi.fhg.de/People/Thomas.Baker/agenda/topic-type-vocabulary

2005-05-20: 13:00-17:30 (4 hours)

08. Process Document - 30-45 min [Stuart, Diane]
http://www.bi.fhg.de/People/Thomas.Baker/agenda/topic-process

09. Using Dublin Core and AskDCMI - 30 min [Diane]
http://www.bi.fhg.de/People/Thomas.Baker/agenda/topic-userguide

10. Madrid meeting [Tom]
http://www.bi.fhg.de/People/Thomas.Baker/agenda/topic-madrid

11. Definitions and Labels - 1 hour? - or Madrid... [Andy]
http://www.bi.fhg.de/People/Thomas.Baker/agenda/topic-definitions-labels

12. Encoding scheme types - 1 hour? - or Madrid... [Andy]
http://www.bi.fhg.de/People/Thomas.Baker/agenda/topic-encoding-scheme-types

Namespace Policy for the Dublin Core Metadata Initiative (DCMI)

Editor: Andy Powell
Editor: Harry Wagner
Contributor: Stuart Weibel
Contributor: Tom Baker
Contributor: Tod Matola
Contributor: Eric Miller
Date Issued: 2001-10-26
Identifier: http://dublincore.org/documents/2001/10/26/dcmi-namespace/
Replaces: http://dublincore.org/documents/2001/09/17/dcmi-namespace/
Is Replaced By: Not Applicable
Latest version: http://dublincore.org/documents/dcmi-namespace/
Status of document: This is a DCMI Recommendation
Description of document: An XML namespace [XML-NAMES] is a collection of names, identified by a URI reference [RFC2396], that are used in XML documents as element types and attribute names. The use of XML namespaces to uniquely identify metadata terms allows those terms to be unambiguously used across applications, promoting the possibility of shared semantics. DCMI adopts this mechanism for the identification of all DCMI terms.

This document specifies the conventions used for identifying current and future DCMI namespaces. All DCMI recommendations that make use of namespaces will conform to this recommendation.

Glossary:

The following are defined terms in this document:

- **DCMI term**
  A *DCMI term* is a DCMI element, a DCMI qualifier or term from a DCMI-maintained controlled vocabulary. Each *DCMI term* is defined in a *DCMI recommendation* and is identified by a Uniform Resource Identifier (URI) within a *DCMI namespace*.
- **DCMI namespace**
  A *DCMI namespace* is a collection of *DCMI terms*. Each *DCMI namespace* is identified by a URI.
- **DCMI recommendation**
  A *DCMI recommendation* is a human-readable document that may define one or more *DCMI terms*.
- **DCMI term declaration**
  A *DCMI term declaration* is the machine-processable representation of one or more *DCMI terms*, expressed in a schema language.

I. Introduction

An XML namespace [XML-NAMES] is a collection of names, identified by a URI reference [RFC2396], that are used in XML documents as element types and attribute names. The use of XML namespaces to uniquely identify metadata terms allows those terms to be unambiguously used across applications, promoting the possibility of shared semantics. DCMI adopts this mechanism for the identification of all DCMI terms.

This document specifies the conventions used for identifying current and future DCMI namespaces. All DCMI recommendations that make use of namespaces will conform to this recommendation.

II. Namespace URIs used by the DCMI

The URI of the namespace for all DCMI elements that comprise the Dublin Core Metadata Element Set, Version 1.1 [DCMES] is:

```
http://purl.org/dc/elements/1.1/
```

The URI of the namespace for all DCMI elements and DCMI qualifiers (other than those elements defined in the Dublin Core Metadata Element Set, Version 1.1 above) is:

```
http://purl.org/dc/terms/
```

The URI of the namespace for DCMI terms defined in the DCMI Type Vocabulary [DCMI-TYPE] is:

```
http://purl.org/dc/dcmitype/
```

Therefore, the three currently approved DCMI namespace URIs are:

http://purl.org/dc/elements/1.1/	Dublin Core Metadata Element Set, Version 1.1 (15 elements)
http://purl.org/dc/terms/	DCMI elements and DCMI qualifiers (other than those elements defined in the Dublin Core Metadata Element Set, Version 1.1 above)
http://purl.org/dc/dcmitype/	DCMI terms in the DCMI Type Vocabulary (a DCMI controlled vocabulary)

All DCMI namespace URIs will resolve to a machine-processable DCMI term declaration for all the terms within that namespace.

The URI for each DCMI term is constructed by appending the term *name* to the namespace URI for that term. For example:

```
http://purl.org/dc/elements/1.1/title
```

is the URI for the Title element in the Dublin Core Metadata Element Set, Version 1.1,

```
http://purl.org/dc/terms/extent
```

is the URI for the Extent qualifier in the Dublin Core Qualifiers recommendation [DCQ] and

```
http://purl.org/dc/dcmitype/Image
```

is the URI for the Image term in the DCMI Type Vocabulary. Each DCMI term can be so identified.

All future DCMI namespace URIs (additional DCMI controlled vocabularies for example) will conform to this pattern:

```
http://purl.org/dc/namespace_label/
```

III. Policy concerning classes of changes to DCMI terms

Changes to DCMI terms or term declarations will occur from time to time for a variety of reasons. Such changes have varying implications for DCMI namespaces. The following classes of changes are identified along with examples and associated implications for namespaces.

In all cases, any changes to DCMI terms or term declarations will result in an update to the versioning information carried in the DCMI recommendation and/or DCMI term declaration associated with that term.

A. Minor editorial errata

Errors of spelling, punctuation, or other clerical mistakes discovered in DCMI recommendations and/or DCMI term declarations will be corrected without a comment period, following notification to the DCMI Usage Board [DCMI-USAGE], as long as, in the judgment of the DCMI Directorate, there are no implications for negative impact on users or applications that rely on those DCMI term declarations.

Correction of minor editorial errata will result in no changes in DCMI namespace URIs.

B. Substantive editorial errata

Errors of substance discovered in DCMI recommendations and/or DCMI term declarations will trigger public notification of the correction to the DC-General mailing list [DC-GENERAL]. Errors that, in the judgment of the DCMI Directorate, compromise the immediate usefulness or accuracy of DCMI metadata systems will be corrected immediately (for example, an incorrect URL to a resource external to DCMI). Others will be corrected following a 14-day public comment period to assure that changes do not adversely effect systems or applications which rely on the DCMI namespace infrastructure.

Correction of substantive editorial errata will result in no changes in DCMI namespace URIs.

C. Semantic changes in DCMI terms

Changes of definitions within DCMI recommendations and/or DCMI term declarations will be reflected in the affected DCMI recommendation and/or DCMI term declaration. If, in the judgment of the DCMI Directorate, such changes of meaning are likely to have substantial impact on either machine processing of DCMI terms or the functional semantics of the terms, then these changes will be reflected in a change of name or namespace for the DCMI term or terms in question. The URIs for any new DCMI namespaces resulting from such changes will conform to the DCMI namespace pattern defined above.

D. Addition of DCMI term declarations to existing DCMI namespaces

New terms will occasionally be added to existing DCMI namespaces. Addition of DCMI terms to existing namespaces will not trigger changes in namespace URIs.

IV. Persistence Policy

The DCMI recognizes that people and applications depend on the persistence of formal documents and machine processable schemas that have been made publicly available. In particular, the stability of namespace URIs for metadata terms is critical to interoperability over time. Thus, the wide promulgation of this set of URIs dictates that they be maintained to support legacy applications that have adopted them.

V. Justification

Two significant issues were raised during the development of this policy. Firstly, that DCMI namespace URIs should indicate the *category* of DCMI terms within that namespace. For example, it was proposed that different DCMI namespaces might be used to partition DCMI elements from DCMI qualifiers, or to indicate that a particular term was originally defined by a particular community or within a particular domain. Secondly, that all DCMI namespace URIs should carry versioning information (for example a date stamp) that would be updated as terms within the namespace change.

On the first issue it was considered that the *category* of DCMI terms was not necessarily persistent. For example, terms defined initially by the education community might subsequently become useful to other communities. Associating particular URIs with particular categories of terms was not felt to be helpful to the long-term stability of DCMI namespaces or the URIs of DCMI terms within those namespaces.

On the second issue it was again considered that embedding versioning information within the namespace URI was unlikely to be helpful to the long-term stability of DCMI namespaces or the URIs of DCMI terms within those namespaces. Rather, it was felt that versioning information should be carried within the DCMI recommendations and/or DCMI term declarations associated with DCMI namespaces and terms.

Finally it should be noted that, although the 15 elements currently within the `http://purl.org/dc/elements/1.1/` namespace could have been redefined within the `http://purl.org/dc/terms/` namespace, it was considered that the widespread existing usage of the former namespace URI mitigated against any change. Furthermore, the existing use of the `purl.org` domain for that namespace URI prompted its use for all DCMI namespace URIs.

References

[XML-NAMES]
Namespaces in XML, W3C Recommendation, 14 January 1999
http://www.w3.org/TR/REC-xml-names

[RFC2396]
IETF (Internet Engineering Task Force) RFC 2396: Uniform Resource Identifiers (URI): Generic Syntax, eds. T. Berners-Lee, R. Fielding, L. Masinter. August 1998.

[DCMES]
Dublin Core Metadata Element Set, Version 1.1: Reference Description
http://dublincore.org/documents/1999/07/02/dces/

[DCMI-TYPE]
DCMI Type Vocabulary, DCMI Recommendation, 11 July 2000
http://dublincore.org/documents/dcmi-type-vocabulary/

[DCQ]
Dublin Core Qualifiers
http://dublincore.org/documents/dcmes-qualifiers/

[DCMI-USAGE]
DCMI Usage Board
http://www.dublincore.org/usage/

[DC-GENERAL]
DC-General mailing list
http://www.jiscmail.ac.uk/lists/dc-general.html

Metadata associated with this resource: http://dublincore.org/documents/dcmi-namespace/index.shtml.rdf

DCMI Abstract Model

Creator: Andy Powell
UKOLN, University of Bath, UK
Mikael Nilsson
KMR Group, CID, NADA, KTH (Royal Institute of Technology), Sweden
Ambjörn Naeve
KMR Group, CID, NADA, KTH (Royal Institute of Technology), Sweden
Pete Johnston
UKOLN, University of Bath, UK

Date Issued: 2005-03-07
Identifier: http://dublincore.org/documents/2005/03/07/abstract-model/
Replaces: http://dublincore.org/documents/2005/01/31/abstract-model/
Is Replaced By: Not applicable
Latest Version: http://dublincore.org/documents/abstract-model/
Status of Document: This is a DCMI Recommendation.
Description of Document: This document describes an abstract model for DCMI metadata descriptions.

Table of contents

1. Introduction

This document specifies an abstract model for DCMI metadata [DCMI]. The primary purpose of this document is to provide a reference model against which particular DC encoding guidelines can be compared. To function well, a reference model needs to be independent of any particular encoding syntax. Such a reference model allows us to gain a better understanding of the kinds of descriptions that we are trying to encode and facilitates the development of better mappings and translations between different syntaxes.

This document is primarily aimed at the developers of software applications that support Dublin Core metadata, people involved in developing new syntax encoding guidelines for Dublin Core metadata and those people developing metadata application profiles based on the Dublin Core.

2. DCMI abstract model

The abstract model of the *resources* being described by DCMI metadata *descriptions* is as follows:

- Each *resource* has zero or more *property/value pairs*.
- Each *property/value pair* is made up of one *property* and one *value*.
- Each *value* is a *resource* (the physical or conceptual entity that is associated with a *property* when it is used to describe a *resource*).
- Each *resource* may be a member of one or more *classes*.
- Each *property* and *class* has some declared *semantics*.
- Each *class* may be related to one or more other *classes* by a refines (sub-class) relationship (where the two *classes* share some *semantics* such that all *resources* that are members of the *sub-class* are also members of the related *class*).
- Each *property* may be related to one or more other *properties* by a refines (sub-property) relationship (where the two *properties* share some *semantics* such that whenever a *resource* is related to a *value* by the *sub-property*, it follows that the *resource* is also related to that same *value* by the *property*).

The abstract model of DCMI metadata *descriptions* is as follows:

- A *description* is made up of one or more *statements* (about one, and only one, *resource*) and zero or one *resource URI* (a URI reference that

identifies the *resource* being described).

- Each *statement* instantiates a *property/value pair* and is made up of a *property URI* (a URI reference that identifies a *property*), zero or one *value URI* (a URI reference that identifies a *value* of the *property*), zero or one *vocabulary encoding scheme URI* (a URI reference that identifies the *class* of the *value*) and zero or more *value representations* of the *value*.
- The *value representation* may take the form of a *value string* or a *rich representation*.
- Each *value string* is a simple, human-readable string that is a representation of the *resource* that is the *value* of the *property*.
- Each *value string* may have an associated *syntax encoding scheme URI* that identifies a *syntax encoding scheme*.
- Each *value string* may have an associated *value string language* that is an ISO language tag (e.g. en-GB).
- Each *rich representation* is some *marked-up text*, an image, a video, some audio, etc. or some combination thereof that is a representation of the *resource* that is the *value* of the *property*.
- Each *value* may be the subject of a separate *related description*.

The italicized words and phrases used above are defined in the terminology section below. A number of things about the model are worth noting:

- A *related description* describes a related *resource* and is therefore not part of the *description* - for example, a *related description* may provide metadata about the person that is the creator of the described *resource*.
- *Syntax encoding schemes* are also known as 'datatypes' in some contexts.
- Each *resource* may be a member of one or more *classes*. Note that where the *resource* is a *value*, the *class* is referred to as a *vocabulary encoding scheme*.
- In DCMI metadata *descriptions*, the *class* of the *resource* being described is normally indicated by the *value* of the DC Type *property*.

The DCMI abstract model for resources and descriptions is represented as UML class diagrams [UML] in figures 1 and 2.

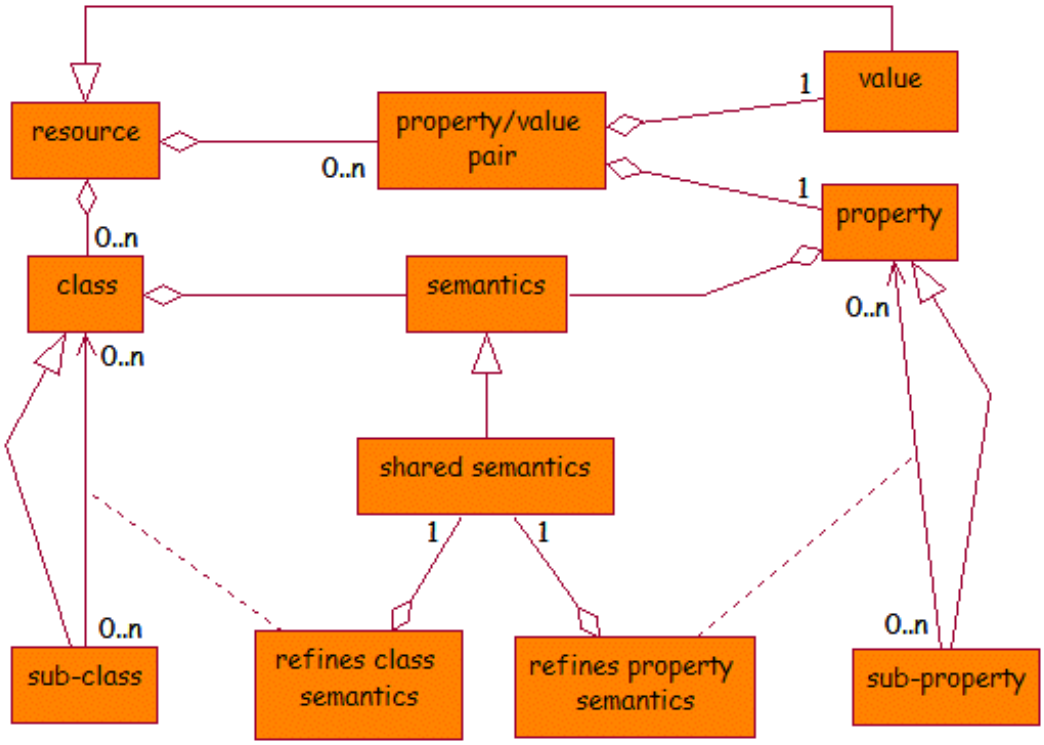

Figure 1 - the DCMI resource model

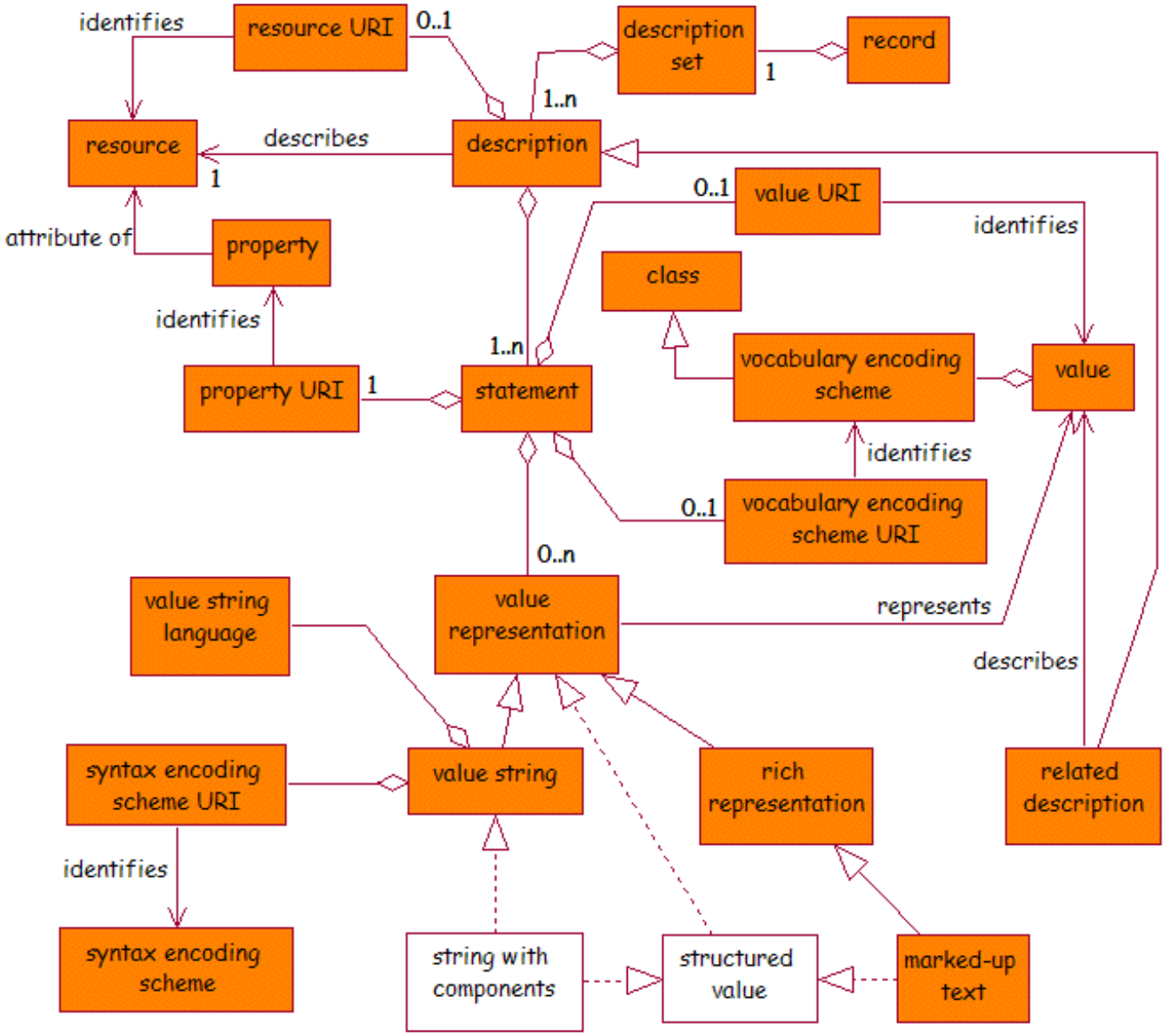

Figure 2 - the DCMI description model

Readers that are not familiar with UML class diagrams should note that lines ending in a block-arrow should be read as 'is' or 'is a' (for example, 'a *vocabulary encoding scheme* is a *class*') and that lines starting with a block-diamond should be read as 'contains a' or 'has a' (for example, 'a *statement* contains a *property URI*'). Other relationships are labeled appropriately. The classes represented by the clear boxes are not mentioned explicitly in the textual description of the abstract model above but are discussed in Appendix A. Note that the UML modeling used here shows the abstract model but is not intended to form a suitable basis for the development of DCMI software applications.

3. Descriptions, description sets and records

The abstract model described above indicates that each DCMI metadata *description* describes one, and only one, resource. This is commonly referred to as the one-to-one principle.

However, real-world metadata applications tend to be based on loosely grouped sets of *descriptions* (where the described *resources* are typically related in some way), known here as *description sets*. For example, a *description set* might comprise *descriptions* of both a painting and the artist. Furthermore, it is often the case that a *description set* will also contain a *description* about the *description set* itself (sometimes referred to as 'admin metadata' or 'meta-metadata').

Description sets are instantiated, for the purposes of exchange between software applications, in the form of metadata *records*, according to one of the DCMI encoding guidelines (XHTML meta tags, XML, RDF/XML, etc.) [DCMI-ENCODINGS].

This document defines a *description set* and a DCMI metadata *record* as follows:

- A *description set* is a set of one or more *descriptions* about one or more *resources*.
- A DCMI metadata *record* is a *description set* that is instantiated according to one of the DCMI encoding guidelines (XHTML meta tags, XML, RDF/XML, etc.)

4. Values

A DCMI metadata *value* is the physical or conceptual entity that is associated with a *property* when it is used to describe a *resource*. For example, the *value* of the DC Creator *property* is a person, organization or service - a physical entity. The *value* of the DC Date *property* is a point (or range) in time - a conceptual

entity. The *value* of the DC Coverage *property* may be a geographic region or country - a physical entity. The *value* of the DC Subject *property* may be a concept - a conceptual entity - or a physical object or person - a physical entity. Each of these entities is a *resource*.

The *value* may be identified using a *value URI*; the *value* may be represented by one or more *value strings* and/or *rich representations*; the *value* may have some *related descriptions* - but the *value* is a *resource*.

5. Dumb-down

The notions of 'simple DC' and 'qualified DC' are widely used within DCMI documentation and discussion fora. This document does not present a definitive view of what these phrases mean because their usage is somewhat variable. However, in general terms, the phrase 'simple DC' is used to refer to DC metadata that does not make any use of *encoding schemes* and *element refinements* and in which each statement only contains a *value string* while the phrase 'qualified DC' is used to refer to metadata that makes use of all the features of the abstract model described here.

The process of translating qualified DC into simple DC is normally referred to as 'dumbing-down'. The process of dumbing-down can be separated into two parts: property dumb-down and value dumb-down. Furthermore, each of these processes can be approached in one of two ways. Informed dumb-down takes place where the software performing the dumb-down algorithm has knowledge built into it about the *property* relationships and *values* being used within a specific DCMI metadata application. Uninformed dumb-down takes place where the software performing the dumb-down algorithm has no prior knowledge about the *properties* and *values* being used.

Based on this analysis, it is possible to outline a 'dumb-down algorithm' matrix, shown below:

	Element dumb-down	**Value dumb-down**
Uninformed	Discard any *statement* in which the *property URI* identifies a *property* that isn't in the Dublin Core Metadata Element Set [DCMES].	Use *value URI* (if present) or *value string* as new *value string*. Discard any *related descriptions* and *rich representations*. Discard any *encoding scheme URIs*.
Informed	Recursively resolve sub-property relationships until a recognised *property* is reached and substitute the *property URI* of that *property* for the existing *property URI* in the *statement*. If no recognised *property* is reached, then discard the *statement*. (In many cases, this process stops when a *property* is reached that is not an *element refinement*.)	Use knowledge of any *rich representations*, *related descriptions* or the *value string* to create a new *value string*.

Note that software should make use of the DCMI *term* declarations represented in RDF schema language [DC-RDFS], the DC XML namespace URIs [DC-NAMESPACES] and the appropriate DCMI encoding guidelines (XHTML meta tags, XML, RDF/XML, etc.) [DCMI-ENCODINGS] to automate the resolution of sub-property relationships.

In cases where software is dumbing-down a *description set* containing multiple *descriptions*, it may either generate several 'simpler' *descriptions* (one per *description* in the original *description set*) or a single 'simple' description (in which case it will have to determine which is the 'primary' *description* in the original *description set*). This is an application-specific decision.

6. Encoding guidelines

Particular encoding guidelines (HTML meta tags, XML, RDF/XML, etc.) [DCMI-ENCODINGS] do not need to encode all aspects of the abstract model described above. However, DCMI recommendations that provide encoding guidelines should refer to the DCMI abstract model and indicate which parts of the model are encoded and which are not. In particular, encoding guidelines should indicate the mechanism by which *resource URIs* and *value URIs* are encoded. Note that the abstract model does **not** indicate that a *value string* with an associated `http://purl.org/dc/terms/URI` *syntax encoding scheme* should be treated as a *value URI* or *resource URI*. Encoding guidelines should provide an explicit mechanism for encoding these features of the model. Encoding guidelines should also indicate whether any *rich representations* or *related descriptions* associated with a *statement* are embedded within the *record* or are encoded in a separate *record* and linked to it using a URI reference.

Appendices B, C and D below provide a summary comparison between the abstract model and the RDF/XML, XML and XHTML encoding guidelines.

7. Terminology

This document uses the following terms:

class
: A *class* is a group containing members that have attributes, behaviours, relationships or semantics in common; a kind of category.

class URI
: A *class URI* is a URI reference that identifies a *class*.

description
: A *description* is made up of one or more *statements* about one, and only one, *resource*.

description set
: A *description set* is a set of one or more *descriptions* about one or more *resources*.

element
: Within DCMI, *element* is typically used as a synonym for *property*. However, it should be noted that the word element is also commonly used to refer to a structural markup component within an XML document.

element refinement
: An *element refinement* is a *property* of a *resource* that shares the meaning of a particular DCMI *property* but with narrower semantics. Since *element refinements* are *properties*, they can be used in metadata *descriptions* independently of the *properties* they refine. In DCMI practice, an *element refinement* refines just one parent DCMI *property*.

encoding scheme
: *Encoding scheme* is the generic name for *vocabulary encoding scheme* and *syntax encoding scheme*.

encoding scheme URI
: The generic name for a *vocabulary encoding scheme URI* or a *syntax encoding scheme URI*.

marked-up text
: A string that contains HTML, XML or other markup (for example TeX) and that is associated with the *value* of a *property*.

property
: A *property* is a specific aspect, characteristic, attribute, or relation used to describe *resources*.

property URI
: A *property URI* is a URI reference that identifies a single *property*.

property/value pair
: A *property/value pair* is the combination of a *property* and a *value*, used to describe a *resource*.

qualifier
: *Qualifier* was the generic name used for the *terms* that are now usually referred to specifically as *element refinements* or *encoding schemes*.

record
: A *record* is a *description set* that is instantiated according to one of the DCMI encoding guidelines (XHTML meta tags, XML, RDF/XML, etc.)

related description

A *related description* is a *description* of a *resource* that is related to the *resource* being described.

resource

A *resource* is anything that has identity. Familiar examples include an electronic document, an image, a service (e.g., "today's weather report for Los Angeles"), and a collection of other *resources*. Not all *resources* are network "retrievable"; e.g., human beings, corporations, concepts and bound books in a library can also be considered *resources*.

resource URI

A *resource URI* is a URI reference that identifies a single *resource*.

rich representation

Some *marked-up text*, an image, a video, some audio, etc. (or some combination thereof) that is associated with the *value* of a *property*.

statement

A *statement* is made up of a *property URI* (a URI reference that identifies a *property*), zero or one *value URI* (a URI reference that identifies a *value* of the *property*), zero or one *vocabulary encoding scheme URI* (a URI reference that identifies the *class* of the *value*) and zero or more *value representations* of the *value*.

structured value

Structured value is the generic name for the following:

- A *value string* that contains machine-parsable component parts (and which has an associated *syntax encoding scheme* that indicates how the component parts are encoded within the string).
- Some *marked-up text*.
- A *related description*

syntax encoding scheme

A *syntax encoding scheme* indicates that the *value string* is formatted in accordance with a formal notation, such as "2000-01-01" as the standard expression of a date.

syntax encoding scheme URI

A *syntax encoding scheme URI* is a URI reference that identifies a *syntax encoding scheme*. For all DCMI recommended *encoding schemes*, the URI reference is constructed by concatenating the name of the *encoding scheme* with the `http://purl.org/dc/terms/` namespace URI.

term

The generic name for a *property* (i.e. *element* or *element refinement*), *vocabulary encoding scheme*, *syntax encoding scheme* or concept taken from a controlled vocabulary (concept space).

term URI

The generic name for a URI reference that identifies a *term*.

value

A *value* is the physical or conceptual entity that is associated with a *property* when it is used to describe a *resource*.

value URI

A *value URI* is a URI reference that identifies the *value* of a *property*.

value representation

A *value representation* is a surrogate for (i.e. a representation of) the *value*.

value string

A *value string* is a simple string that represents the *value* of a *property*. In general, a *value string* should not contain any *marked-up text*.

value string language

The *value string language* indicates the language of the *value string*.

vocabulary encoding scheme

A *vocabulary encoding scheme* is a *class* that indicates that the *value* of a *property* is taken from a controlled vocabulary (or concept-space), such as the Library of Congress Subject Headings.

vocabulary encoding scheme URI

A *vocabulary encoding scheme URI* is a URI reference that identifies a *vocabulary encoding scheme*. For all DCMI recommended *encoding schemes*, the URI reference is constructed by concatenating the name of the *encoding scheme* with the `http://purl.org/dc/terms/` namespace URI.

References

DCMI

Dublin Core Metadata Initiative
<http://dublincore.org/>

UML

The Unified Modeling Language User Guide
Grady Booch, James Rumbaugh and Ivar Jacobson, Addison-Wesley, 1998

DCTERMS

DCMI Metadata Terms
<http://dublincore.org/documents/dcmi-terms/>

DCMES

Dublin Core Metadata Element Set, Version 1.1: Reference Description
<http://dublincore.org/documents/dces/>

DCMI-ENCODINGS

DCMI Encoding Guidelines
<http://dublincore.org/resources/expressions/>

DC-RDFS

DCMI term declarations represented in RDF schema language
<http://dublincore.org/schemas/rdfs/>

DC-NAMESPACES

Namespace Policy for the Dublin Core Metadata Initiative (DCMI)
<http://dublincore.org/documents/dcmi-namespace/>

Acknowledgements

Thanks to Tom Baker, Rachel Heery, the members of the DC Usage Board and the members of the DC Architecture Working Group for their comments on previous versions of this document.

Appendix A - A note about structured values

This appendix discusses 'structured values', as they are used in DC metadata applications at the time of writing.

Many existing applications of DC metadata have attempted to encode relatively complex 'value representations' (i.e. representations that are not just a simple string). These attempts have been loosely referred to as 'structured values'. It is possible to identify a number of different kinds of structured values that have been commonly used. Four are enumerated below. The first two of these are recommended by the DCMI, in the sense that there are existing encoding schemes that define values that conform to these definitions of structured values. The latter two are not currently recommended, but it is likely that they are in fairly common usage across metadata applications worldwide.

Labelled strings

These are strings that contain explicitly labeled components. Examples of this kind of structured value include:

DCSV
: and the various DCMI syntax encoding schemes built on it - Period, Point and Box. An example of the use of DCSV in Period is:

```
<meta name="dcterms:temporal"
      scheme="dcterms:Period"
      content="start=Cambrian period; scheme=Geological timescale; name=Phanerozoic Eon;" />
```

vCard
: for example:

```
<meta name="dc:creator"
      content="BEGIN:VCARD\nORG:University of Oxford\nEND:VCARD\n" />
```

Note that vCard is not currently a DCMI recommended encoding scheme.

Unlabeled strings

These are strings that contain implicit components within the string, i.e. the components are determined based solely on their position within the string. Examples of this kind of structured value include:

W3CDTF
: the date-time format used within most DC metadata. For example:

```
<meta name="dc:date"
      scheme="dcterms:W3CDTF"
      content="2003-06-10" />
```

Marked-up text

These are strings containing 'presentational' or other markup, for example adding paragraph breaks, superscripts or chemical/mathematical markup to a dc:description. It is possible to characterize various kinds of markup as follows:

- Markup based on a version of HTML.
- Markup based on other XML-based languages such as CML and MathML.
- Non-XML markup languages such as TeX.

Related resource descriptions

These are metadata descriptions that describe a second resource (i.e. not the resource being described by the DC description). For example, a related description associated with the value of dc:creator could contain a complete description of the resource author (including birthday, eye-colour and favourite beverage if desired!).

In the past, 'related resource descriptions' have tended to be encoded using XML, vCard (see above) or by inventing multiple 'refinements' of DCMES properties (for example DC.Creator.Address). The RDF/XML encoding of DC (see below) provides us with a more thorough modeling of related metadata records through the use of multiple linked nodes in an RDF graph.

Summary

The categories outlined above are not watertight and there are certainly overlaps between them. For example, labeled strings can be viewed as a type of non-XML markup language. In addition, there will be cases where marked-up text (e.g. MathML) can be viewed as a related resource description.

Nevertheless, the purpose of the categorization used here is to try and analyze existing usage of complex metadata structures within current DC metadata applications. In the context of the abstract model proposed here, all the types of structured values outlined above form part of the DCMI abstract model:

- A labeled string should be treated as a *related description* (though it should be noted that DCSV and the various DCMI syntax encoding schemes built on it - Period, Point and Box - are currently encoded as *value strings* with an appropriate *syntax encoding scheme*).
- An unlabeled string should be treated as a *value string* with an appropriate *syntax encoding scheme*.
- Marked-up text should be treated as a *rich representation*.
- A related resource description should be treated as a *related description*.

Appendix B - The abstract model and RDF

This appendix discusses the relationship between the DCMI abstract model and the Resource Description Framework (RDF).

RDF currently provides DCMI with the richest encoding environment of the available encoding syntaxes. It is therefore worth taking a brief look at how the abstract model described here compares with the RDF model.

Note that the intention here is not to provide a full and detailed description of how to encode DC metadata records in RDF. Instead, three simple examples of the use of DC in RDF are considered.

Example 1: dc:creator

Figure 3 shows a simple RDF graph (and the RDF/XML document that represents it). The graph shows a resource with a single property (dc:creator). The *value* of the property is a second (blank) node, representing the creator of the resource. This second blank node has several properties, used to describe the creator, and an rdfs:label property that is used to provide the *value string* for the dc:creator property.

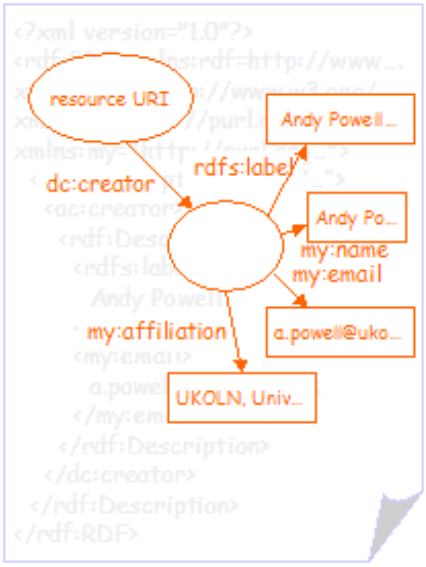

Figure 3

Figure 4 shows the same information separated into two graphs. In this case the *related description* that describes the creator has been more clearly separated from the description of the resource by moving it into a separate RDF/XML document. In order to do this, the node representing the *value* has been assigned a *value URI*, allowing the two nodes in the two RDF/XML documents to be treated as representing the same thing.

The *related description* in the second RDF/XML document is linked to the first using the rdfs:seeAlso property and the URI of the RDF/XML document. Note that it is not strictly necessary to separate the two graphs in this way; it is perfectly valid to represent the second graph as a sub-graph of the first, as shown in figure 3. However, for the purposes of this document, the two graphs have been separated in order to more clearly differentiate the *description* from the *related description*. In some cases it will be good practice to facilitate this separation anyway. For example, in order to serve the second graph from a directory service of some kind.

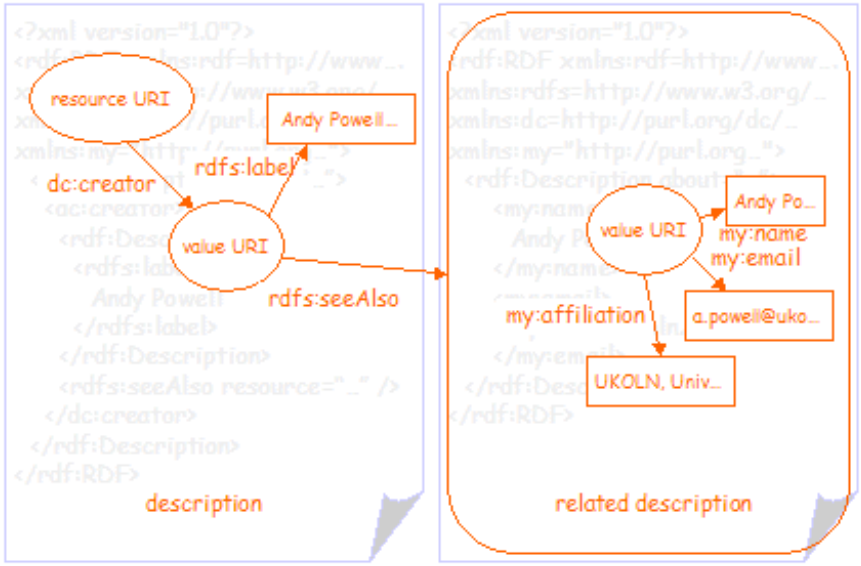

Figure 4

Example 2: dc:subject

Figure 5 shows a second simple RDF graph (and the RDF/XML document that represents it). The graph shows a resource with a single property (dc:subject). The *value* of the property is a second (blank) node, representing the subject of the resource. This second blank node has an rdfs:label property that is used to provide the *value string* for the dc:subject property, an rdf:value property that is used to provide the classification scheme notation and an rdf:type property to provide the encoding scheme URI.

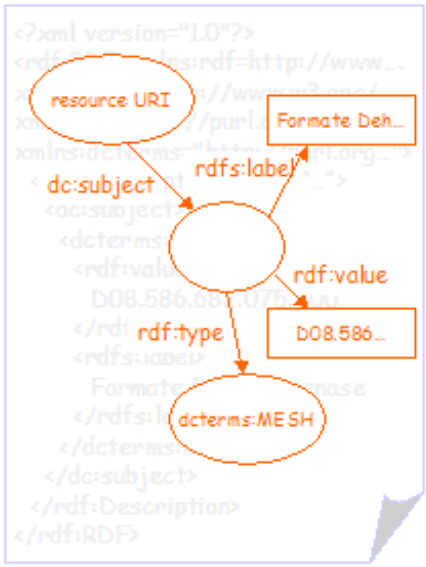

Figure 5

Figure 6 shows the same information separated into two graphs. In this case the *related description* that describes the subject has been more clearly separated from the description of the resource by moving it into a separate RDF/XML document. In order to do this, the node representing the *value* has been assigned a *value URI*, allowing the two nodes in the two RDF/XML documents to be treated as representing the same thing.

The *related description* in the second RDF/XML document is linked to the first using the rdfs:seeAlso property and the URI of the RDF/XML document. Note that it is not strictly necessary to separate the two graphs in this way; it is perfectly valid to represent the second graph as a sub-graph of the first, as shown in figure 5. However, for the purposes of this document, the two graphs have been separated in order to more clearly differentiate the *description* from the *related description*. In some cases it will be good practice to facilitate this separation anyway. For example, in order to serve the second graph from a terminology service of some kind.

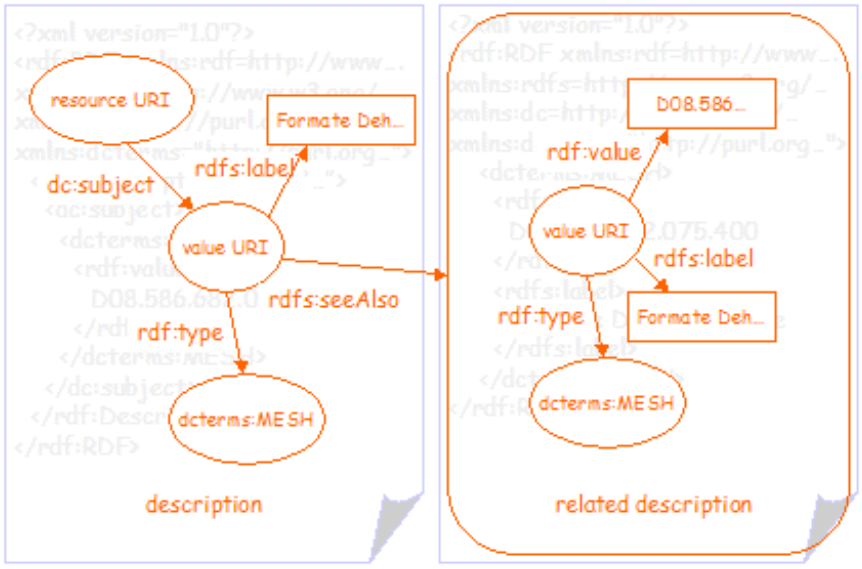

Figure 6

Example 3: dc:description

Figure 7 shows a third simple RDF graph (and the RDF/XML document that represents it). The graph shows a resource with a single property (dc:description). The *value* of the property is a second (blank) node with an rdfs:label property that is used to provide the *value string* for the dc:description property. The second node also has an rdfs:seeAlso property that links to a *rich representation* - in this case some HTML *marked-up text* that provides a richer representation of the description.

Note that it is possible to embed the *marked-up text* within a single RDF graph (using rdf: parseType="Literal"). However, this is not shown here.

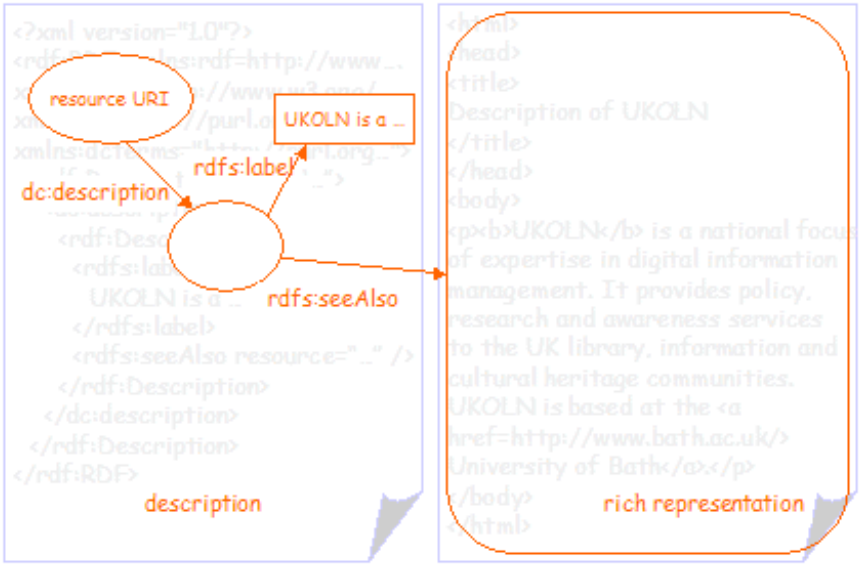

Figure 7

Summary

By re-visiting the second figure from example 2 (figure 6) it is possible to layer the terminology used in the abstract model above over the RDF graph.

Almost all aspects of the DCMI abstract model are supported by the RDF encoding guidelines though, at the time of writing, some issues about how best to handle *description sets* still need to be resolved.

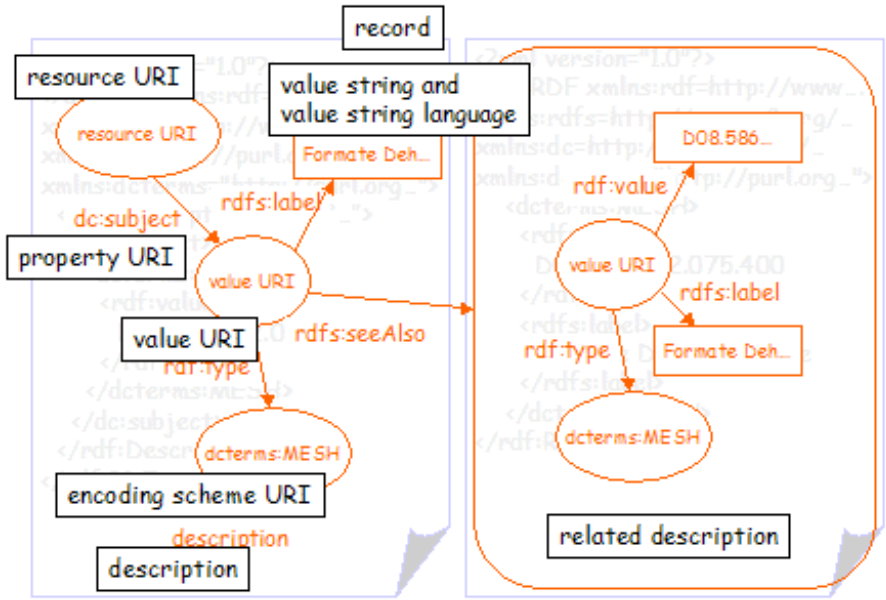

Figure 8

Appendix C - The abstract model and XML

This appendix compares the DCMI abstract model with the Guidelines for implementing Dublin Core in XML DCMI recommendation.

Simple DC

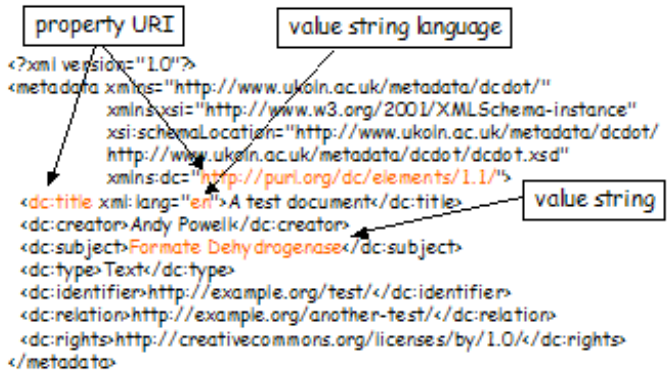

Figure 9

Figure 9 shows an example simple DC description encoded according to the XML guidelines above. The example shows how the encoding supports the *property URI*, *value string* and *value string language* aspects of the DCMI abstract model. It should be noted that all the *values* that are encoded in this syntax are represented by *value strings*, even those that look, to the human reader, as though they are URIs.

Qualified DC

Figure 10

Figure 10 shows an example qualified DC description encoded according to the XML guidelines above. This example shows how the encoding supports the *property URI*, *value string*, *value string language*, *encoding scheme URI* and *resource class* aspects of the DCMI abstract model. Note also that, although the *resource class* is indicated, the *class URI* is not encoded anywhere in this description.

Summary

The following aspects of the DCMI abstract model are supported by the Guidelines for implementing Dublin Core in XML recommendation:

- *properties*
- *property URIs*
- *value strings*
- *value string languages*
- *encoding schemes*
- *encoding scheme URIs*
- *resource classes*

The following aspects of the DCMI abstract model are not supported:

- *resource URIs*
- *value URIs*
- *rich representations*
- *related descriptions*
- *property*/sub-property relationships
- *resource class URIs*

The following constraints apply:

- Each *property* may have one *value string* (but not more than one).
- *Vocabulary encoding schemes* and *syntax encoding schemes* are handled in exactly the same way.

Note that, at the time of writing, neither *resource URIs* nor *value URIs* can be explicitly encoded in the XML encoding syntax. Although it may be the case that some software applications have chosen to interpret the use of a `http://purl.org/dc/terms/URI` *syntax encoding scheme* as an indication that the URI in the *value string* is a *resource URI* or *value URI*, this is **not** guaranteed to be a correct interpretation of the metadata *record* in all cases.

Appendix D - The abstract model and XHTML

This appendix compares the DCMI abstract model with the Expressing Dublin Core in HTML/XHTML meta and link elements DCMI recommendation.

Simple DC

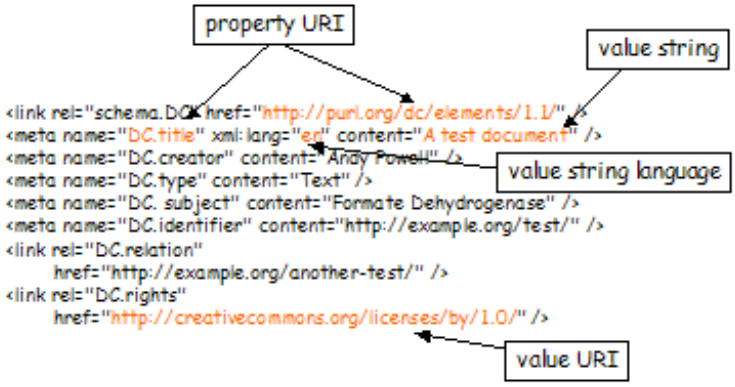

Figure 11

Figure 11 shows an example simple DC description encoded according to the XHTML guidelines above. This example shows how the encoding supports the *property URI*, *value string*, *value string language* and *value URI* aspects of the DCMI abstract model. Again, it should be noted that the value of the DC Identifier *property* represented in this encoding syntax is denoted by a *value string*, even though it looks, to the human reader, as though it is a URI.

Qualified DC

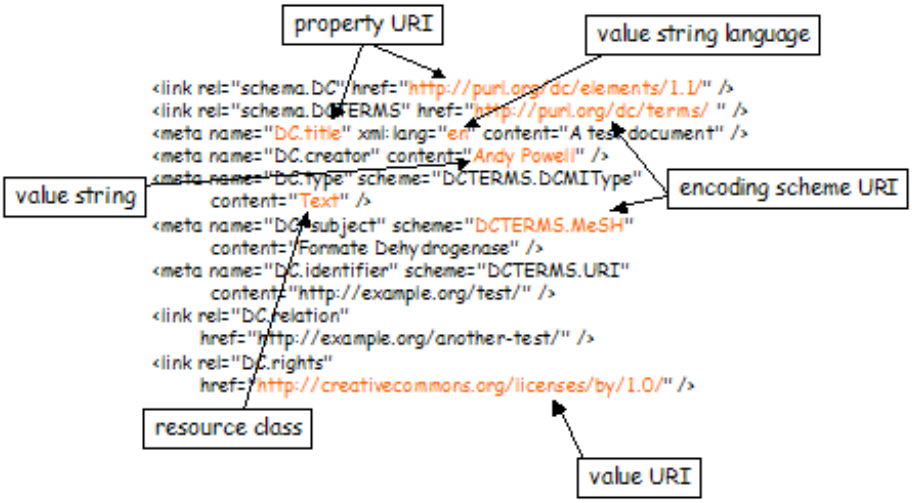

Figure 12

Figure 12 shows an example qualified DC description encoded according to the XHTML guidelines above. This example shows how the encoding supports the *property URI*, *value string*, *value string language*, *value URI*, *encoding scheme URI* and *resource class* aspects of the DCMI abstract model. Note that although the *resource class* is indicated, the *class URI* is not encoded anywhere in this description. Finally, note that although the `http://purl.org/dc/terms/URI` *syntax encoding scheme* means that software can reliably interpret the DC Identifier *value string* as a URI, it should not be interpreted as a *resource URI*.

Summary

The following aspects of the DCMI abstract model are supported by the Expressing Dublin Core in HTML/XHTML meta and link elements DCMI recommendation:

- *properties*
- *property URIs*
- *value strings*
- *value string languages*
- *value URIs*
- *encoding schemes*
- *encoding scheme URIs*
- *resource classes*

The following aspects of the DCMI abstract model are not supported:

- *resource URIs*
- *rich representations*
- *related descriptions*
- *property*/sub-property relationships
- *resource class URIs*

The following constraints apply:

- Each *property* may have one *value string* (but not more than one) or a *value URI* but not both.
- *Vocabulary encoding schemes* and *syntax encoding schemes* are handled in exactly the same way.

Note that, at the time of writing, *resource URIs* cannot be explicitly encoded in the XHTML encoding syntax. However, the *resource URI* may be implicit from the URI of the *resource* into which the *record* is embedded.

DCMI Policy on Naming Terms

Creator: Stuart Weibel
Creator: Thomas Baker
Date Modified: 2004-04-05
Identifier: http://dublincore.org/documents/2004/04/05/naming-policy/
Is Replaced By: Not applicable
Latest Version: http://dublincore.org/documents/naming-policy/
Status of Document: This is a DCMI Recommended Resource.
Description of Document: This document describes the policy followed in assigning Names to DCMI terms, particularly with regard to case sensitivity.

DCMI Term Names

The Dublin Core Metadata Initiative maintains sets of metadata terms. Each metadata term is assigned a *name* -- a character string or "token" that is unique in the context of a particular DCMI term set [DCMI-TERMS]. In accordance with the DCMI Namespace Policy, the *name* of a term is appended to the URI of a *DCMI namespace* to construct a globally unique identifier (URI) for that particular term [DCMI-NAMESPACE]. The typology of DCMI metadata terms is described in the DCMI Grammatical Principles [DCMI-PRINCIPLES].

Case Policy for DCMI Term Names

Names of DCMI Elements and Element Refinements start with lowercase characters but may contain uppercase characters where a term name is comprised of multiple concatenated words. In such cases, the leading character of additional words will be capitalized to improve human-readable clarity. Examples include the names:

creator
audience
isReplacedBy

which are used to derive the following URIs in accordance with the DCMI Namespace Policy:

http://purl.org/dc/elements/1.1/creator
http://purl.org/dc/terms/audience
http://purl.org/dc/terms/isReplacedBy

Names of Encoding Schemes identifying controlled vocabularies by acronym are represented in all uppercase characters. Examples include the names:

DDC
W3CDTF
ISO639-2

which are used to derive the following URIs in accordance with the DCMI Namespace Policy:

http://purl.org/dc/terms/DDC
http://purl.org/dc/terms/W3CDTF
http://purl.org/dc/terms/ISO639-2

Names of Encoding Schemes other than acronyms are represented with a leading uppercase character followed by lowercase characters. Examples include the names:

Period
Box

which are used to derive the following URIs in accordance with the DCMI Namespace Policy:

http://purl.org/dc/terms/Period
http://purl.org/dc/terms/Box

Names of values within a DCMI-maintained controlled vocabulary such as the DCMI Type Vocabulary begin with an uppercase character followed by lowercase, but with subsequent concatenated words in the name capitalized as well. Examples include the names:

InteractiveResource
Collection
Dataset

which are used to derive the following URIs in accordance with the DCMI Namespace Policy:

http://purl.org/dc/dcmitype/InteractiveResource
http://purl.org/dc/dcmitype/Collection
http://purl.org/dc/dcmitype/Dataset

No DCMI Term Names will be assigned that differ from other Names only in regard to case.

Historical note

Since the mid-1990s, Dublin Core has developed in parallel with World-Wide Web technology, and Dublin Core metadata has been deployed using all of the major encoding syntaxes that have evolved. In the transition from HTML to XML, XHTML, and RDF/XML, conventions regarding the naming of metadata elements have consolidated in the wider Web community, and DCMI policy has changed to follow the emerging model of good practice.

In 1998, the Dublin Core element set was published using Names (at the time called Labels) which had a leading uppercase character, e.g. Title and Creator [RFC2413]. In October 2000, the DCMI Advisory Committee decided to change the Names of elements to lowercase in order to bring DCMI practice into line with conventions widely (though not universally) followed in existing Dublin Core applications and in the XML and RDF/XML communities more generally.

Unfortunately, this decision was propagated throughout DCMI documentation only after some delay [DC-ELEMENTS]. In the meantime, the Dublin Core Metadata Element Set had progressed through formal standardization channels for recognition first as NISO Z39.85-2001 and CEN Workshop Agreement CWA 13874, then as ISO 15836 -- with element Names starting in uppercase [ISO15836].

The DCMI Directorate is not aware of any applications or implementations where this inconsistency with regard to the case of element Names has caused any practical problems. For the sake of consistency, however, the Directorate will undertake to amend existing specifications as permitted by their maintenance cycles.

Confusion over case sensitivity is commonplace in Web protocols, and metadata application developers are likely to find many permutations of case in instance data and term declarations. Thus, to paraphrase a dictum from early Web protocol development, applications should:

...be rigorous in what you export, and tolerant in what you import from other applications.

In practice, this suggests that applications are well advised to normalize case when parsing terms for identity comparisons. Prudence mitigates against the use of case to distinguish between alternative identities of related terms in any namespace, and it is DCMI policy that such distinctions not be made within its own namespaces, so it is unlikely that errors would be introduced by normalizing case.

References

[DCMI-TERMS]
http://dublincore.org/documents/dcmi-terms/

[DCMI-NAMESPACE]
http://dublincore.org/documents/dcmi-namespace/

[RFC2413]
http://www.ietf.org/rfc/rfc2413.txt

[DC-ELEMENTS]
http://dublincore.org/documents/2002/10/06/current-elements/

[ISO15836-2003]
http://www.niso.org/international/SC4/n515.pdf

Feedback on this document

The DCMI Directorate welcomes comments and suggestions on this policy: dcmi-feedback@dublincore.org

Metadata associated with this resource: http://dublincore.org/documents/naming-policy/index.shtml.rdf

Title:	**Guidelines for assigning identifiers to metadata terms**
Creator:	Andy Powell UKOLN, University of Bath, UK
Date Issued:	2004-08-01
Identifier:	http://dublincore.org/documents/2004/08/01/term-identifier-guidelines/
Replaces:	
Is Replaced By:	Not applicable
Latest Version:	http://dublincore.org/documents/term-identifier-guidelines/
Status of Document:	This is a DCMI Working Draft.
Description of Document:	This document provides some simple guidelines for assigning identifiers to non-DCMI metadata terms (elements, element refinements, encoding schemes and vocabulary terms).

1. Introduction

The Dublin Core Abstract Model [DC-AM] requires that all terms (elements, element refinements, encoding schemes and controlled vocabulary terms) used in metadata application profiles that are compliant with the model must be assigned a URI reference that identifies the term. A URI reference is a Uniform Resource Identifier with an optional fragment identifier (sometimes known as a URIref) [RFC2396]. An XML namespace [XML-NAMES] is a collection of names, identified by a URI reference that are used in XML documents as element types and attribute names. By convention, all DCMI recommended encodings [DCMI-ENCODINGS] use a concatenation of an XML namespace URI reference and the term name to provide a mechanism for encoding the term URI reference. The use of XML namespaces and URI references to uniquely identify metadata terms allows those terms to be unambiguously used across applications, promoting the possibility of shared semantics. As indicated in the DCMI Namespace Policy [DCMI-NAMESPACE], DCMI has adopted this mechanism for the identification of all DCMI terms.

This document provides some simple guidelines for assigning URI references to metadata terms in non-DCMI namespaces. This includes non-DCMI elements, element refinements, encoding schemes and controlled vocabulary terms.

Although these guidelines are mainly intended for metadata application profiles that conform with the Dublin Core Abstract Model, it is hoped that they are generic enough that they may be useful in the context of other metadata applications as well.

2. Guidelines

All metadata terms must be assigned a URI reference. The use of fragment identifiers in the URI references used to identify metadata terms is optional and is left to the discretion of the implementor.
Should something be inserted here about current W3C TAG thinking that conceptual resources should be identified using URI references that have a fragment identifier?

For the purposes of encoding, the term URI reference may be partitioned into an XML namespace URI reference

and the term name. Note that, for convenience, it is commonly the case that XML namespace URI references end with either a '#' (hash) or '/' (slash) character.

Groups of related terms (for example, all the terms within a controlled vocabulary) should be assigned URI references within the same XML namespace.

All XML namespace and term URI references should resolve to human and/or machine-readable descriptions of the namespace or term.

Any valid URI reference [RFC2396] may be used to identify a metadata term. However, the use of a registered URI scheme is recommended [URI-SCHEMES].

All XML namespace and term URI references should be assigned with the intention of them being unique and persistent. This means that the URI reference must not be used to identify anything else and that it should be expected to last as long as the Internet.
Is this a reasonable definition of 'persistent'?

3. Strategies for assigning URI references

Four simple strategies for assigning URI references to metadata terms are described below.

3.1 Using service or project URLs

Where a term is created within the context of a particular project, service or other initiative, the use of a project or service-specific URL may be appropriate. This is probably the simplest strategy in terms of ease of assignment and resolution. However, it is also the most prone to lack of persistence.

Example 1: `http://myservice.org/terms/price`
: An existing service is delivered using the `myservice.org` DNS domain name. The service creates a new property called `price` for use in its metadata application profile. The service defines an XML namespace URI reference within its existing URL space (`http://example.org/terms/`) and therefore assigns the term the following URI reference: `http://example.org/terms/price`.

Example 2: `http://myproject.org/metadata/vocabs/color#Red`
: A project Web-site is delivered using the `myproject.org` DNS domain name. The project team build up a new controlled vocabulary of colors for use within their metadata application profile. They define an XML namespace URI reference within their existing URL space (`http://myproject.org/metadata/vocabs/color#`). For the vocabulary term `Red`, the term URI reference is therefore `http://myproject.org/metadata/vocabs/color#Red`

Notice that example 1 defines a metadata property while example 2 defines a term within a controlled vocabulary. Remember that in example 2 it will probably also be necessary to define an encoding scheme name for the vocabulary itself, for example `http://myproject.org/metadata/terms/Color`.

3.2 Using PURLs

A similar approach, but one that is likely to offer more persistent URI references, is to use PURLs [PURL]. A PURL is a Persistent Uniform Resource Locator. Functionally, a PURL is a URL. However, instead of pointing directly to the location of an Internet resource, a PURL points to an intermediate resolution service. This provides a level of resilience aginst changes in project or service URLs. The use of PURLs to identify metadata terms has already been adopted by a number of metadata-related initiatives such as DCMI itself and RDF Site Summary (RSS) 1.0 [RSS10].

Example 1: `http://purl.org/rss/1.0/link`
: RDF Site Summary is a lightweight multipurpose extensible metadata description and syndication format. The core metadata terms used by RSS are declared within an XML namespace (`http://purl.org/rss/1.0/`). For example, the property called `link` has been assigned the URI reference `http://purl.org/rss/1.0/link`. Other terms are declared within separate groupings, known in RSS as modules. Each module makes use of one or more separate XML namespaces.

Example 2: `http://purl.org/rdn/terms/dateReviewed`
: The UK JISC-funded Resource Discovery Network has developed a small metadata application profile in order to describe the status of its catalogue records. One of the new terms in the application profile is called `dateReviewed`. All the new terms have been defined within an RDN XML namespace (`http://purl.`

org/rdn/terms/). Therefore, the URI reference assigned to the dateReviewed property is http://purl.org/rdn/terms/dateReviewed.

Note that in example 1, the RSS implementors have chosen to embed a version number into the XML namespace URI reference. This allows them to use the same term name within a new XML namespace in future versions of the application profile. This has advantages in some scenarios. However, implementors should be cautious when using this technique because it may result in URI references being assigned to new terms that have the same semantics as existing terms.

3.3 Using "info" URIs

The "info" URI scheme provides a *"mechanism for assigning URI references to information assets that have identifiers in public namespaces"* but that do not have an appropriate existing URI scheme [INFO-URI-SPEC]. The phrase 'information assets' includes all the metadata terms discussed here. Thus, it is appropriate to consider assigning "info" URIs to metadata terms.

Example 1: info:ddc/22/eng//004.678

The terms that make up the Dewey Decimal Classification [DEWEY] have been assigned "info" URIs such that info:ddc/22/eng// can be considered to be an XML namespace URI reference and "004.678" can be considered to be a Dewey term name. Thus the URI reference that has been assigned to that term is info:ddc/22/eng//004.678. Note that the information asset identified by this term is in the English-language Dewey Decimal Classifications (22nd Ed.) and is the classification "Internet".

Note that, somewhat confusingly, the draft "info" URI specification uses different terminology from that used here. In the terminology of the specification, ddc is the *"info URI namespace component"* and 22/eng//004.678 is the *"info URI identifier component"*.

Note also that "info" URIs can not be resolved using current Web browsers (i.e. by using a simple HTTP GET request), though they can be looked-up in the "info" URI registry [INFO-REGISTRY].

At the time of writing, "info" was not a registered URI scheme.

3.4 Using xmlns.com

xmlns.com provides a network space for simple Web namespace management. *"The rationale for registering xmlns.com was to secure a short, memorable domain suitable for naming concepts for use in RDF and XML vocabularies"* [XMLNS]. The FOAF vocabulary [FOAF] uses xmlns.com to provide an XML namespace URI reference for its terms.

Example 1: http://xmlns.com/foaf/0.1/firstName

The firstName term within the FOAF vocabulary uses the http://xmlns.com/foaf/0.1/ XML namespace URI reference and has been assigned the URI reference http://xmlns.com/foaf/0.1/firstName.

Note that, at the time of writing, the status and ownership of the xmlns.com domain was slightly unclear and it is therefore not possible to be sure of the long term persistence of URI references based on this domain.
Is this a fair comment?

4. Conclusions

In general, all terms used in metadata application profiles must be assigned a URI reference before they can be used in the encoding syntaxes recommended by DCMI. It is recommended that implementors assign URI references to terms following the guidelines provided here. Of the four strategies for assigning URI references to terms listed in this document, the use of PURLs is recommended for the identification of elements, element refinements and encoding schemes and the use of PURLs or "info" URIs is recommended for the identification of terms in controlled vocabularies.
Do people agree with this conclusion? Is more justification for this conclusion required?

References

[DC-AM]

DCMI Abstract Model, 4 February 2004
http://www.ukoln.ac.uk/metadata/dcmi/abstract-model/

[XML-NAMES]
Namespaces in XML, W3C Recommendation, 14 January 1999
http://www.w3.org/TR/REC-xml-names

[RFC2396]
IETF (Internet Engineering Task Force) RFC 2396: Uniform Resource Identifiers (URI): Generic Syntax, eds. T. Berners-Lee, R. Fielding, L. Masinter. August 1998.

[DCMI-ENCODINGS]
DCMI Encoding Guidelines
http://dublincore.org/resources/expressions/

[DCMI-NAMESPACE]
Namespace Policy for the Dublin Core Metadata Initiative (DCMI), 26 October 2001
http://dublincore.org/documents/dcmi-namespace/

[URI-SCHEMES]
Uniform Resource Identifier (URI) SCHEMES
http://www.iana.org/assignments/uri-schemes

[PURL]
PURLS
http://purl.org/

RDF Site Summary 1.0
http://purl.org/rss/1.0/spec

[INFO-URI-SPEC]
The "info" URI Scheme for Information Assets with Identifiers in Public Namespaces, 9 July 2004
http://info-uri.info/registry/docs/drafts/draft-vandesompel-info-uri-02.txt

[DEWEY]
Dewey Decimal Classification
http://www.oclc.org/dewey/

[INFO-REGISTRY]
"info" URI registry
http://info-uri.info/

[XMLNS]
xmlns.com
http://xmlns.com/

[FOAF]
FOAF Vocabulary Specification
http://xmlns.com/foaf/0.1/

Metadata associated with this resource: http://dublincore.org/documents/term-identifier-guidelines/index.shtml.rdf

CEN

WORKSHOP

AGREEMENT

CWA 14855

November 2003

ICS 35.100.05; 35.240.60

English version

Dublin Core Application Profile Guidelines

This CEN Workshop Agreement has been drafted and approved by a Workshop of representatives of interested parties, the constitution of which is indicated in the foreword of this Workshop Agreement.

The formal process followed by the Workshop in the development of this Workshop Agreement has been endorsed by the National Members of CEN but neither the National Members of CEN nor the CEN Management Centre can be held accountable for the technical content of this CEN Workshop Agreement or possible conflicts with standards or legislation.

This CEN Workshop Agreement can in no way be held as being an official standard developed by CEN and its Members.

This CEN Workshop Agreement is publicly available as a reference document from the CEN Members National Standard Bodies.

CEN members are the national standards bodies of Austria, Belgium, Czech Republic, Denmark, Finland, France, Germany, Greece, Hungary, Iceland, Ireland, Italy, Luxembourg, Malta, Netherlands, Norway, Portugal, Slovakia, Spain, Sweden, Switzerland and United Kingdom.

EUROPEAN COMMITTEE FOR STANDARDIZATION
COMITÉ EUROPÉEN DE NORMALISATION
EUROPÄISCHES KOMITEE FÜR NORMUNG

Management Centre: rue de Stassart, 36 B-1050 Brussels

Ref. No. CWA 14855:2003 D/E/F

Contents

Foreword

The production of this CEN Workshop Agreement (CWA) was formally accepted as part of the CEN/ISSS Workshop on Metadata for Multimedia Information - Dublin Core (WS/MMI-DC) in the Workshop's plenary meeting on 2002-03-07.

This CWA was agreed upon by the contributing partners in the CEN/ISSS Workshop on MMI-DC, representing a wide mix of interests, including administrations, libraries, on-line education and geographic information systems. The list of company individuals who have supported the document's contents may be obtained from the CEN/ISSS Secretariat.

The CWA was approved by the Workshop’s plenary meeting on 2003-09-08.

The final text of this CWA was submitted to CEN for publication on 2003-09-09.

Introduction

A Dublin Core Application Profile (DCAP) is a declaration specifying which metadata terms an organization, information provider, or user community uses in its metadata. By definition, a DCAP identifies the source of metadata terms used – whether they have been defined in formally maintained standards such as Dublin Core, in less formally defined element sets and vocabularies, or by the creator of the DCAP itself for local use in an application. Optionally, a DCAP may provide additional documentation on how the terms are constrained, encoded, or interpreted for application-specific purposes.

A DCAP is designed to promote interoperability within the constraints of the Dublin Core model and to encourage harmonization of usage and convergence on “emerging semantics" around its edges. Historically, application profiles have emerged out of a need to share local domain- or application-specific refinements of or extensions to Dublin Core within particular application communities without necessarily seeking an extension of the core standard maintained by the Dublin Core Metadata Initiative (DCMI). Application profiles document how implementers use elements from Dublin Core along with elements from other vocabularies, customizing standard definitions and usage guidelines for local requirements [HEERY].

In practice, application profiles are created for a wide range of purposes: to document the semantics and constraints used for a set of metadata records (“instance metadata”); to help communities of implementers harmonize metadata practice among themselves; to identify emerging semantics as possible candidates for formal standardization; as guides for semantic crosswalks and format conversions; as specifications for formal encoding structures such as Document Type Definitions (DTDs); for interpreting or presenting legacy or proprietary metadata in terms of widely-understood standards; or for documenting the rules and criteria according to which a set of metadata was created. Application profiles often represent “work in progress”, providing foci for ongoing efforts to incrementally improve and clarify a body of shared metadata semantics within a particular user community.

In the absence of guidelines, creators of application profiles have hitherto invented a wide range of presentation formats. The present document distils the salient features of many existing profiles into a format that is as concise and simple as possible, yet as precise and detailed as is sometimes necessary to support the various uses identified above.

Semantic interoperability – the ultimate purpose of documents such as DCAPs – is a longer-term goal to be pursued as metadata vocabularies and related enabling technologies mature over time. In their current form, DCAPs are designed to document metadata usage in a normalized form that will lend itself to translation into common models, such as RDF, that can be processed by machines to automate such interoperability.

Machine-understandable representations will achieve this goal to the extent that metadata terms can be referenced using stable, well-documented identifiers. As discussed below, the practice of identifying metadata terms with Uniform Resource Identifiers (URIs) is currently gaining momentum. Maintaining a DCAP over time, then, may involve improving its precision incrementally by identifying its terms with URIs as the URIs become available; this is referred to here as the Principle of Appropriate Identification.

In the meantime, these guidelines aim at the more modest aim of providing system developers and information specialists with a normalized and readable view of Dublin-Core-based metadata models. A DCAP should include enough information to be of optimal usefulness for its intended audience – a Principle of Readability – even if this entails the redundant inclusion of information, which, in a formal system of machine-processable schemas, might otherwise be fetched dynamically from external sources.

Given the flexibility of presentation required by the Principle of Readability, no assumption is made that DCAPs will be convertible into future machine-understandable forms without the use of ad-hoc heuristics or manual intervention. Creators of DCAPs should bear in mind that a normalized form of documentation cannot itself address the deeper problems of interoperability in a world with a diversity of underlying metadata models – problems which will continue to challenge the metadata community as a whole, and the Dublin Core Metadata Initiative in particular, for the foreseeable future.

1 Scope

The present document gives guidance on how information should be structured and presented in Dublin Core Application Profiles. Principles and concepts underlying DCAPs as declarative metadata constructs are defined and explained.

The guidelines do not mandate a particular document format for DCAPs. DCAPs may be presented as plain text files or as Web pages, word-processing files, PowerPoint, or indeed as ink on paper. By providing a consistent presentation structure for such documents, these guidelines aim at making it easier for people to understand what others are doing in their metadata. The guidelines mandate enough structure to ensure that DCAPs will be convertible as straightforwardly as possible into expressions that use schema languages, such as RDF, for automatic processing by machines. In this sense, a normalized documentational form for DCAPs is a first step towards the more ambitious and long-term goal of automating semantic interoperability across a broad diversity of information sources.

2 Definitions

Dublin Core Application Profile (DCAP): A DCAP is a declaration specifying which metadata terms an organization, information provider, or user community uses in its metadata and how those terms have been customized or adapted to a particular application. By definition, a DCAP is based in part on Dublin Core and follows DCMI Grammatical Principles [DCMI-PRINCIPLES]. A DCAP consists of a Descriptive Header and one or more Term Usages.

DCMI Grammatical Principles: As maintained by the Dublin Core Metadata Initiative, DCMI grammatical principles specify a typology of metadata terms – Elements, Element Refinements, Encoding Schemes, and Vocabulary Terms – along with their interrelationships and functions [DCMI-PRINCIPLES]. A DCAP is based on the simple model of a resource described with a flat set of properties. This is consistent with DCMI grammatical principles, which do not themselves specify more elaborate models.

Descriptive Header: A Descriptive Header places the DCAP into an interpretive context by specifying, at a minimum, a Title, Creator, Date, Identifier, and Description for the DCAP. An optional Preamble may comment on any technical or stylistic conventions followed in the DCAP.

Term Usage: A Term Usage is a description of a metadata term, which, at a minimum, identifies a metadata term in accordance with the Principle of Appropriate Identification by using one or more identifying attributes – Term URI, Defined By, Name, Label – as described in Section 3. Optionally, a Term Usage may also describe or annotate a term in more detail by providing additional definitional attributes, relational attributes, or constraints, as described in Section 4.

Principle of Appropriate Identification: The Principle of Appropriate Identification dictates that metadata terms be identified as precisely as possible. As established in the so-called CORES Resolution of December 2002, the preferred method for identifying a metadata term is to cite its Uniform Resource Identifier (URI). All DCMI metadata terms are identified with URIs, and URIs are currently being assigned to the terms of other major semantic standards such as MARC21 and IEEE/LOM [CORES-RESOLUTION]. Whenever such URIs are available they should be cited as an attribute of a Term Usage (its “Term URI”). Terms to which URIs have not (or not yet) been assigned should be identified using other attributes as appropriate, as described in Section 3.

Principle of Readability: The Principle of Readability dictates that a DCAP should include enough information in Term Usages to be of optimal usefulness for the intended audience of the DCAP – even if this entails the redundant inclusion of information which, in a formal system of machine-processable schemas, might otherwise be fetched dynamically from external sources. Conversely, the Principle of Readability allows unused attributes simply to be omitted from display.

3 Identifying terms with appropriate precision

Application profiles serve to clarify who is declaring and maintaining the metadata semantics that a group wants to share. This section describes how a metadata term used in a Term Usage can be identified with appropriate precision (the Principle of Appropriate Identification).

At present, the preferred method for identifying a metadata term is to cite its Uniform Resource Identifier (URI) if such is available. A URI is "a compact string of characters for identifying an abstract or physical resource" constructed according to a generic and flexible syntax [URI]. The World Wide Web Consortium has promoted the notion that “All important resources should be identified by a URI” [WEBARCH] and has specifically promoted the use of URIs for identifying metadata elements. In the CORES Resolution of December 2002, the maintainers of seven leading metadata standards – Dublin Core, IEEE/LOM, DOI, CERIF, MARC21, ONIX, and GILS -- pledged to assign URIs to their elements and to articulate policies for the persistence of those URIs [CORES-RESOLUTION]. (Note that a URI, when used to identify a metadata term, often functions as a Web address for accessing information about that term, such as a Web page or machine-processable schema. However, the CORES Resolution does not require that such identifiers resolve to such resources, and URIs that result in “file not found” messages are not necessarily “broken” as identifiers.)

For metadata terms to which a URI has been officially assigned – for example, by DCMI or by another signatory of the CORES Resolution – that URI should be cited in the field “Term URI”. For example, the Dublin Core element “Audience” should be cited as “http://purl.org/dc/terms/audience”. As this form of identification is precise and sufficient on its own, other identifying fields may be left blank:

```
Term URI        http://purl.org/dc/terms/audience
Name            -
Label           -
Defined By      -
```

In accordance with the Principle of Readability, other identifying attributes such as Name and Label could be added here to make the DCAP more “reader-friendly”. If the DCAP is intended as a guide for processing metadata records, it may indeed be necessary to provide a Name (i.e., the string actually used in the metadata records). If the Name or Label of a term are considered more “reader-friendly” as captions for a Term Usage than the Term URI, the order of these attributes may be changed to put these first. See Sections 5.4 (“Attributes copied from external sources”) and 5.2 (“Readability of Term Usages”) for further discussion.

A term that has been declared or documented somewhere but not assigned a URI (as far as one knows) should be identified as precisely as possible by providing its name and pointing to a declarative document or schema in which it has been defined. The declarative document or schema should be cited with URI, Web address, or bibliographic reference in the field "Defined By". The term itself can be cited using either a string identifier or token (in the field “Name”, which by default is assumed to be case-sensitive) or a natural-language label (in the field "Label"), or both, taken from the declarative document or schema:

```
Term URI        -
Name            AttendancePattern
Label           Attendance Pattern
Defined By      http://someones-project.org/schema.html
```

For a term that has not already been defined in any other declarative document, the field Defined By should simply cite the URI of the DCAP itself (as assigned with Identifier in the DCAP Descriptive Header). For example, in a DCAP with the URI "http://my-project.org/profile.html", a new local term called Star Ratings could be defined as follows:

```
Term URI        -
Name            StarRatings
Label           Star Ratings
Defined By      http://my-project.org/profile.html
```

A creator of a DCAP wishing to declare locally coined terms in a way that makes them citable with precision, and thus re-usable by others, may undertake the additional step of assigning them URIs. At present, the technical conventions and "Web etiquette" for naming metadata terms with URIs have yet to establish themselves in common practice, though at a minimum it seems both polite and sensible not to promote new URIs unless it is expected they will be maintained. For the purposes of DCAPs, DCMI itself provides models of practice, and further options are likely to emerge as the CORES Resolution is implemented [DCMI-NAMESPACE, DCMI-TERMS, DCMI-SCHEMAS]. Note that the CORES Resolution itself addresses the use of URIs as identifiers only and is silent on whether the URIs should resolve to informational Web pages or schemas [CORES-RESOLUTION].

4 Attributes of a Term Usage

Attributes for describing the metadata terms "used" in a DCAP are listed below. Note that they are called "attributes" here simply to avoid confusingly recursive formulations such as "terms for describing terms".

Use of Identifying Attributes in Term Usages (see Section 4.1) is governed by the Principle of Appropriate Identification. According to this principle, a Term Usage should use one or more of the four Identifying Attributes to identify a term as precisely as appropriate – i.e., with a formally assigned URI if available, or alternatively by citing a name or label for the term along with a reference to a document, schema, or Web page in which that term is defined.

All of the other attributes of Term Usages are optional and should be used as local needs may dictate. As discussed in Section 5.4, “local” and “source” attributes may be distinguished as necessary.

4.1 Identifying attributes

Term URI	A Uniform Resource Identifier used to identify the term.
Name	A unique token assigned to the term.
Label	A human-readable label assigned to the term.
Defined By	An identifier of a namespace, pointer to a schema, or bibliographic reference for a document within which the term is defined.

4.2 Definitional attributes

Definition	A statement that represents the concept and essential nature of the term.
Comments	Additional information about the term or its application.
Type of term	A grammatical category of the term (e.g., "Element", "Element Refinement", or "Encoding Scheme").

4.3 Relational attributes

Refines	The described term semantically refines the referenced term.
Refined By	The described term is semantically refined by the referenced term.
Encoding Scheme For	The described term, an Encoding Scheme, qualifies the referenced term.
Has Encoding Scheme	The described term is qualified by the referenced Encoding Scheme.
Similar To	The described term has a meaning the same as, or similar to, that of the referenced term.

4.4 Constraints

Obligation	Indicates whether the element is required to always or sometimes be present (i.e., contain a value). Examples include "Mandatory", "Conditional", and "Optional".)
Condition	Describes the condition or conditions according to which a value shall be present.
Datatype	Indicates the type of data that can be represented in the value of the element.
Occurrence	Indicates any limit to the repeatability of the element.

5 Discussion

5.1 Descriptive Headers

By definition, a DCAP consists of a Descriptive Header and one or more Term Usages (see Section 2). The Description Header should include the following:

- A brief description of the DCAP based on Dublin Core. At a minimum, the description should specify a Title, Contributor, Date, Identifier, and Description, as explained in more detail in Annex A. Ideally, the description of the DCAP will elaborate on the context in which the DCAP is intended to be used.

- Optionally, a Preamble for the DCAP should describe any technical or formatting conventions used in the DCAP. For example, if namespace prefixes are used in the Name field (see Section 5.6), these prefixes should be documented here. The Preamble can also cite Web pages or schemas in which the terms used in the DCAP are documented and defined so that such information does not need to be repeated in the "Defined By" field of each individual Term Usage; see the example in Section 6.1.1.

5.2 Readability of Term Usages

By default, each term cited in a DCAP should be described with its own Term Usage – a table with a full set of attributes on the left and attribute values on the right. In accordance with the Principle of Readability, however, the intended use of a DCAP may dictate a different presentational style: while DCAPs intended for use by software developers will need to be explicit and detailed, DCAPs intended primarily as informational documents for human consumption can (and often should) be much terser. The following are several ways in which Term Usages may be formatted for readability:

- Instead of creating a separate Term Usage for every Element Refinement and Encoding Scheme used in an application, such terms may simply be cited in the attributes Refined By or Has Encoding Scheme of the Term Usages of the Elements to which they refer (see Section 5.3). Note that this terser style does not support the addition of usage notes, local definitions, or annotations, for which a full Term Usage must be used.

- Attributes not needed for Term Usages can simply be omitted. At one extreme, a Term Usage might legitimately consist of just a Name and a Term URI; see the example in Section 6.1.2).

- The order of attributes presented in Section 4 is significant only for the usability of DCAPs as documents – not for future machine-processable representations of DCAPs. Authors of DCAPs may therefore change the order of attributes in the interest of readability, though they should bear in mind that any such changes may make it more difficult for people to compare two DCAPs visually. For examples of how Name (in boldface) is placed before Term URI, see Section 6.1.2 and DCMI-TERMS.

- In the interest of readability, it might make sense to describe Elements, Element Refinements, and Encoding Schemes with different subsets of relevant attributes. Indeed, these different types of terms might be grouped under separate sections of the DCAP document. (For example, see how Elements and Element Refinements are separated from Encoding Schemes in the document "DCMI Metadata Terms" [DCMI-TERMS].)

- In the interest of readability and of future machine-parsability, attributes should be repeated when necessary (as opposed to listing multiple values for a single attribute).

5.3 "Using" Element Refinements, Encoding Schemes, and Vocabulary Terms

5.3.1 Using Vocabulary Terms

According to DCMI Grammatical Principles, a Vocabulary Term is a member of a controlled vocabulary of values, and a controlled vocabulary of values (as a whole) is named by an Encoding Scheme [DCMI-PRINCIPLES].

In general, it is not the role of application profiles to declare controlled vocabularies of values, either in the sense of creating lists of potential values or in the sense of giving that list (as a whole) a name and URI. Sets of Vocabulary Terms are most appropriately declared in separately citable documents external to a DCAP.

However, if the creator of a DCAP merely wishes to specify a short list of possible values (e.g., "Animal, Vegetable, or Mineral"), these can be simply listed in a "Comment" field.

5.3.2 Using Encoding Schemes

There are three ways to cite Encoding Schemes in a DCAP:

- The most concise way is to use the attribute Has Encoding Scheme (or Comment) for a blanket reference to a set of encoding schemes documented elsewhere. The DCAP for Resource Discovery Network, for example, simply cites "RDN Subject Encoding Schemes" and gives a URL where the list of those encoding schemes may be found; see the example in Section 6.1.2.

- A more precise way is to use the attribute Has Encoding Scheme, repeated as necessary, to cite each Encoding Scheme by its URI (or by Name and URI); for example, see the example in Section 6.3.2.

- In addition to citing Encoding Schemes in the Has Encoding Scheme attribute of Elements, creators of DCAPs may want to describe Encoding Schemes in stand-alone Term Usages in order to annotate their usage, for example by specifying a Datatype, Occurrence, or Local Definition. The attribute Encoding Scheme For points back to the Element or Element Refinement qualified.

5.3.3 Using Element Refinements

The options for Element Refinements are analogous to those for Encoding Schemes:

- Statements such as "all terms in Vocabulary D can be used as element refinements for Contributor" can be simply recorded in a Refined By attribute (or as a Comment).

- Element Refinements can be cited one-by-one using the attribute Refined By; see the example in Section 6.3.2.

- Element Refinements can additionally be described in separate Term Usages.

5.4 Attributes copied from external sources

Ideally, application profiles would be dynamically up-dated with information on the terms they use directly from schemas on the Web and this information would be integrated with local annotations into a "one-stop" document for the convenience of users. The use of machine-understandable DCAPs may some day make this possible.

In the meantime, however, creators of DCAPs who wish to include definitions or other such information from original source documents in their Term Usages have no choice but to copy that information from the source. While the Principle of Readability specifically permits this, authors of DCAPs should bear in mind that copied information, if not maintained, can go out of alignment with the official source.

Where information copied from external sources is supplied, this fact should be reported in the Preamble as described in Section 5.1. Where it is necessary to distinguish in a DCAP between attributes defined locally and attributes copied from an external source, the DCAP should establish its own document-internal

convention, such as distinguishing between a Local Definition and a Source Definition; see the example in Section 6.3.2.

5.5 Types of Comments

Past creators of application profiles for Dublin Core have invented many types of annotation, the most popular of which have been Notes, Best Practice, Usage, Scope, Open Questions, Examples, Purpose, Guidelines, and Don't Confuse With. While the present guidelines lump all of the above into a generically named Comments field, creators of DCAPs may wish to repeat this field with different labels as needed. The needs of future machine processing do not now seem to dictate tighter uniformity in this area.

5.6 Term URIs versus Qualified Names

In the sense intended here, Qualified Names are names of metadata terms that are “qualified” with a prefix standing for a namespace with which the terms are associated (a “namespace prefix”). For example, the Dublin Core element “Title” is sometimes referenced in metadata records and usage documentation using a namespace prefix such as “DC.” or “dc:” as in “DC.Title” or “dc:title”. As straightforward as this citation method may seem, it is based on assumptions about the nature of “namespace” that cannot be assumed to hold across different application environments (e.g., HTML versus RDF versus relational databases) or metadata communities (e.g., for citing elements from standards other than Dublin Core), and at any rate it presupposes an additional mechanism or declaration for associating prefixes with the proper namespaces.

For such reasons, it is far better to cite an element with a full URI – indeed, this is the only method supported by the CORES Resolution and by DCMI policy [CORES-RESOLUTION, DCMI-NAMESPACE]. According to the Principle of Appropriate Identification followed in these guidelines, a Term URI must be cited when available.

On the other hand, long strings such as “http://purl.org/dc/elements/1.1/title” are not very readable and may be misunderstood by the average reader of a DCAP. In accordance with the Principle of Readability, therefore, the author of a DCAP may choose to use qualified names (e.g., “dc:title”) in the “Name” field – as long as any prefixes used are explained in the Preamble of the DCAP, and as long as any available Term URIs are cited as well.

5.7 Declaring new elements

There is nothing to restrain the creator of a DCAP from creating new URIs as identifiers for locally coined metadata terms. For reasons discussed above in Section 3, one should perhaps pause for reflection before taking this step, and if URIs are declared, this step should perhaps be documented separately and not embedded “in passing” into a DCAP full of Term Usages. Any URIs declared for use in a DCAP might best be formed by following the DCMI algorithm and concatenating the URL of the DCAP (e.g., "http://myproject.org/profile/") and the Name of the term (e.g., "starRatings") into a single string (e.g., "http://myproject.org/profile/starRatings") [DCMI-NAMESPACE]. Other models for forming URIs as identifiers for metadata elements are emerging with the implementation of the CORES Resolution [CORES-RESOLUTION].

5.8 Documenting grouped or nested metadata elements

In order to be usable across a diversity of application environments, Dublin Core was designed as a flat set of attributes for describing a resource. In implementation practice, however, Dublin Core elements may be embedded in more elaborate models that group or nest the elements in locally specific ways.

In the absence of a clear and widely accepted data model beyond that of the flat set of attributes, however, applications for integrating metadata from many different sources may be able only to extract and interpret the metadata in terms of Simple Dublin Core, losing any application-specific modelling context. An application designer wishing to document nesting or grouping constructs in a DCAP will need to extend the guidelines described here in order to do so and should bear in mind that documenting such constructs will not in itself guarantee that they will be understood or correctly processed by other applications.

5.9 Documenting unorthodox practices

For reasons both of history and of expedience, a significant number of applications have metadata based on interpretations of the Dublin Core model that are unsound from the standpoint of today's grammatical principles. For example, an application may use CreatorDateOfBirth – an element representing the birth date of a creator of a resource that does not, however, semantically "refine" Creator as its name may imply.

Rather than incorrectly asserting "CreatorDateOfBirth" to be an Element Refinement refining http://purl.org/dc/elements/1.1/creator, the Term Usage in the DCAP should simply record the local name of the element and identify the URI of the DCAP itself as its source. For example, if the DCAP itself is identified by "http://myproject.org/profile/2003/03/17/", the Term Usage should declare the following, leaving empty any fields (such as "Term URI" and "Refines") that would make incorrect assertions about the element:

```
Term URI        -
Local Name      CreatorDateOfBirth
Defined By      http://my-project.org/profile.html
Refines         -
```

Whether "errors" such as "CreatorDateOfBirth" will be of negative consequence for interoperability will depend on how they are interpreted and used in the context of particular applications. The analytical effort involved in creating a DCAP is in effect an important first step towards putting such applications onto a more interoperable foundation.

6 Examples

6.1 UK Resource Discovery Network OAI Application Profile

6.1.1 Descriptive Header

Title	RDN OAI Application Profile
Contributor	Andy Powell
Date	2003-03-23
Identifier	URL for this document – to be assigned
Description	This document expresses the application profile established by the Resource Discovery Network (RDN) to be used by RDN partners for harvesting of records using the Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH). The Application Profile is expressed according to guidelines published by the CEN/ISSS [Reference]. Full user documentation for the Application Profile, together with associated XML schemas, is available at http://www.rdn.ac.uk/oai/rdn_dc/. All Dublin Core terms are fully documented at http://www.dublincore.org/documents/dcmi-terms/.

6.1.2 A Term Usage

Name	Subject
Term URI	http://purl.org/dc/elements/1.1/subject
Has Encoding Scheme	DC Subject Encoding Schemes
Has Encoding Scheme	RDN Subject Encoding Schemes
Comment	RDN Subject Encoding Schemes are available from http://www.rdn.ac.uk/publications/cat-guide/subject-schemes/
Obligation	Recommended

6.1.3 Commentary

The DCAP for the UK Resource Discovery Network is formatted in the tersest possible style [RDN]. Note in particular the following:

- The Descriptive Header puts the DCAP into a specific usage context.
- Only the attributes actually used in a given Term Usage are shown. Indeed, most of the Term Usages in this DCAP consist of just a Name and a Term URI.
- Instead of listing all of the DCMI- and RDN-maintained Encoding Schemes as separate Has Encoding Scheme entries (or as separate Term Usages), this DCAP uses shorthand references to “DC Subject Encoding Schemes” and “RDN Subject Encoding Schemes”. Pointers to documentation are given in the Descriptive Header (for the former) and in a Comment field of the Term Usage (for the latter).
- This DCAP increases the readability of Term Usages by listing the Name, in boldface, before the Term URI. This option is discussed in Section 5.2.

6.2 Renardus Application Profile

6.2.1 Descriptive Header

Title	Renardus Application Profile
Contributor	Metadata Working Group SUB Göttingen
Date	18-04-2002
Identifier	http://renardus.sub.uni-goettingen.de/renap/renap.html
Description	Cross search and cross browse European quality controlled subject gateways.

6.2.2 A Term Usage

Term URI	http://purl.org/dc/elements/1.1/language
Name	Language
Label	Language
Defined By	-
Definition	-
Comments	Renardus: The language code is the ISO 639-2, three-letter code. SUB will provide a mapping between the two letter and three letter language code but this will also be found on the LoC site – ISO 639-2: http://lcweb.loc.gov/standards/iso639-2/englangn.html
Type of term	Element
Refines	-
Refined By	-
Encoding Scheme For	-
Has Encoding Scheme	http://purl.org/dc/terms/ISO639-2
Similar To	-
Obligation	Mandatory
Condition	-
Datatype	String
Occurrence	Repeatable

6.2.3 Commentary

The DCAP for the Renardus Project has been formatted in a somewhat more verbose style [RENARDUS]. Note in particular:

- The DCAP uses its own URL as an identifier.

6.3 UK e-Government Metadata Standard Application Profile

6.3.1 Descriptive Header

Addressee	Metadata Working Group, Interoperability Working Group
Contributor	Drafted by Interoperability and Metadata Analyst, Office of the e-Envoy, Cabinet Office, UK farah.ahmed@e-envoy.gsi.gov.uk
Contributor	Metadata Working Group
Coverage.spatial	UK
Creator	Senior Policy Advisor, Interoperability and Metadata, Office of the e-Envoy, Cabinet Office, UK
Date.issued	2003-08-05
Description	The elements and refinements that provide the structure for metadata used by the UK public sector, designed to complement the e-GMS.
Format	Text/MS Word 2003
Identifier	http://purl.oclc.org/NET/e-GMS-AP_v1
Language	Eng
Publisher	Office of the e-Envoy, Cabinet Office, UK. govtalk@e-envoy.gsi.gov.uk
Rights.copyright	http://www.hmso.gov.uk/docs/copynote.htm Crown Copyright
Source	http://purl.oclc.org/NET/e-GMS_v2
Status	Version 1.0 For publication
Subject	Metadata
Subject.category	Information management
Title	UK e-government metadata standard application profile version 1

6.3.2 A Term Usage

Term URI	http://purl.org/dc/elements/1.1/date
Defined by	http://purl.org/dc/elements/1.1/
Name	Date
Label	Date
Source Definition	A date associated with an event in the life cycle of the resource.
Source Comment	-
Local Definition	-
Local Comment: Purpose	To enable the user to find the resource by limiting the number of search hits according to a date, e.g. the date the resource was made available.
Local Comment: Notes	Dates need to appear in a format that is recognisable to people all over the world, and that can be interpreted by computer software. The W3C format allows accurate searching, and makes it clear which is the year, month or day. The format is ‘ccyy-mm-dd’, where ‘ccyy’ is the year, ‘mm’ is the month and ‘dd’ the day. When the time is also needed, add ‘hh:mm’, where ‘hh’ is the hour (using the 24 hour clock), ‘mm’ is minutes. More about this notation can be found at http://www.w3.org/TR/NOTE-datetime.

Local Comment: Not to be confused with	Coverage – Date refers to dates relevant to the information resource itself, not the information held within the resource. For example, for a document about the civil service in the 18th century, put '18th century' in Coverage and put the date published in Date.
Type of term	Element
Refines	-
Refined by	http://www.govtalk.gov.uk/terms/dateAcquired
Refined by	http://purl.org/dc/terms/dateAccepted
Refined by	http://purl.org/dc/terms/Available
Refined by	http://purl.org/dc/terms/dateCopyrighted
Refined by	http://purl.org/dc/terms/created
Refined by	http://purl.org/dc/terms/issued
Refined by	http://purl.org/dc/terms/dateSubmitted
Refined by	http://purl.org/dc/terms/valid
Refined by	http://purl.org/dc/terms/modified
Refined by	http://www.govtalk.gov.uk/terms/cutOffDate
Refined by	http://www.govtalk.gov.uk/terms/dateDeclared
Refined by	http://www.govtalk.gov.uk/terms/dateClosed
Refined by	http://www.govtalk.gov.uk/terms/nextVersionDue
Refined by	http://www.govtalk.gov.uk/terms/updatingFrequency
Encoding Scheme For	-
Has Encoding Scheme	http://purl.org/dc/terms/W3CDTF
Has Encoding Scheme	http://purl.org/dc/terms/Point
Similar To	-
Constraints	The value must always be taken from the specified encoding scheme, with the exception of the 'updatingFrequency' refinement.
Obligation	Mandatory
Condition	A value must be given either for the unqualified date or at least one date refinement
Datatype	-

6.3.3 Commentary

The DCAP for the UK e-Government Metadata Standard is formatted in the most detailed and specific possible style [EGMS]. While this results in a significantly longer document than the DCAPs for RDN and Renardus, such specificity may be helpful to developers of applications that need to create or process metadata based on the DCAP. Note in particular the following:

- Encoding Schemes and Element Refinements are listed using repeated fields in the Term Usage of the Element to which they refer. In addition, each Encoding Scheme and Element Refinement is also described in its own Term Usage, which allows information about each of them, such as Definition and Constraints, to be recorded in the DCAP as well.

- The Term Usage marks information coming from outside sources: the "Source Definition" copies the definition of Date from DCMI documentation, while the "Local Comment" supplies usage information local to this DCAP.

Annex A: Metadata describing a DCAP

A DCAP should itself be described with Dublin Core metadata, either in a header or in a separate metadata record. At a minimum, this description should include:

Title	A name for the Application Profile.
Contributor	A creator or maintainer of the Profile.
Date	The date of last modification.
Identifier	An unambiguous reference to the Profile. Best practice is to provide a URL by which a copy of the document or schema can be retrieved over the Web.
Description	A concise description of the Profile. As appropriate, the description should elaborate on the context and purposes in which the DCAP is intended to be used; the organizations or individuals involved in its development; any arrangements, policies, or intentions regarding the future development and maintenance of the DCAP; or technical characteristics of the instance metadata or database described.

Annex B: Options for machine-interpretable DCAPs

DCAPs can be expressed in machine-interpretable schema languages, and such machine-interpretable schemas can be manipulated by software applications. This CWA does not give detailed recommendations on how such schemas should be structured, as a number of issues are still open for debate. The scope of this CWA is limited to recommending how application profiles can be expressed as text documents. Future options for machine-interpretable DCAPs are outlined below.

Currently, two schema languages specified by W3C might be considered: XML Schema [XML-SCHEMA] and RDF Schema [RDF-SCHEMA]. The choice of schema language will be influenced by the functionality that the schema is intended to support – for example, whether it is required as a predictable format for data exchange or intended to support inferences about existing metadata. Such different objectives imply different choices between the two schema languages. There has been some discussion on ways to combine XML Schema and RDF Schema to more fully express characteristics of application profiles [HUNTER]. More recently there has been an attempt within the W3C to differentiate RDF Schema as a vocabulary description language and XML Schema as a basis for providing structured data exchange.

An XML schema provides a structured expression that supports validation of instance metadata. In effect, an XML schema provides a document "template" which acts as an exchange format for metadata instances. An XML Schema serves the same function as an XML DTD with additional capability for extensibility and namespace handling.

An RDF schema expresses relationships between terms, providing a data model for expressing the semantics of terms – their properties, classes, and definitions. The underlying RDF data model combined with the use of unique identifiers allows software to infer relationships between terms and perform data aggregation.

RDF Schemas are effective for expressing the semantics of application profiles, whilst XML Schemas are more effective for expressing cardinality, data-typing, and constraints. Possible approaches to the expression of application profiles in RDF have been explored within projects such as SCHEMAS [BAKER] and MEG [MEG-REGISTRY].

Bibliography

[BAKER] Thomas Baker, Makx Dekkers, Rachel Heery, Manjula Patel, Gauri Salokhe, What terms does your metadata use? Application profiles as machine-understandable narratives. Journal of Digital Information 2:2 (November 2001), http://jodi.ecs.soton.ac.uk/Articles/v02/i02/Baker.

[CORES-RESOLUTION] Thomas Baker, Makx Dekkers, Identifying Metadata Elements with URIs: the CORES Resolution. D-Lib Magazine (July 2003), http://www.dlib.org/dlib/july03/baker/07baker.html.

[DC-LIBRARY] Library Application Profile, http://dublincore.org/documents/2002/09/24/library-application-profile/.

[DCMI-NAMESPACE] Andy Powell, Harry Wagner, Stuart Weibel, Tom Baker, Tod Matola, Eric Miller, Namespace policy for the Dublin Core Metadata Initiative, http://dublincore.org/documents/dcmi-namespace/.

[DCMI-PRINCIPLES] DCMI Grammatical Principles, http://dublincore.org/usage/documents/principles/.

[DCMI-SCHEMAS] DCMI Schemas, http://dublincore.org/schemas/.

[DCMI-TERMS] DCMI Metadata Terms, http://dublincore.org/documents/dcmi-terms/.

[EGMS] Office of the e-Envoy – Cabinet Office, UK e-Government Metadata Standard Application Profile Version 1, http://purl.oclc.org/NET/eGMSAPv1.

[HEERY] Rachel Heery, Manjula Patel, Application profiles: mixing and matching metadata schemas, Ariadne 25, September 2000, http://www.ariadne.ac.uk/issue25/app-profiles/intro.html.

[HUNTER] Jane Hunter, Carl Lagoze, Combining RDF and XML Schemas to enhance interoperability between metadata application profiles. WWW10, May 1-5, 2001, Hong Kong, http://www10.org/cdrom/papers/572/index.html.

[MEG-REGISTRY] Rachel Heery, Pete Johnston, Dave Beckett, Damian Steer, The MEG Registry and SCART: Complementary Tools for Creation, Discovery and Re-use of Metadata Schemas. In: Proceedings of the International Conference on Dublin Core and Metadata for e-Communities, 2002. Florence: Firenze University Press, 2002, pp. 125-132, http://www.bncf.net/dc2002/program/ft/paper14.pdf.

[RDF-SCHEMA] Brickley, Dan and Guha, R.V, editors. RDF Vocabulary Description Language 1.0: RDF Schema W3C Working Draft 23 January 2003, http://www.w3.org/TR/rdf-schema/.

[RDN] Powell, Andy, RDN OAI Application Profile, [URL to be assigned].

[RENARDUS] Metadata Working Group SUB Goettingen, Renardus Application Profile, http://renardus.sub.uni-goettingen.de/renap/renap.html.

[URI] T. Berners-Lee, R. Fielding, L. Masinter, Uniform Resource Identifiers (URI): Generic Syntax, August 1998, http://www.ietf.org/rfc/rfc2396.txt.

[WEBARCH] Ian Jacobs, ed., Architecture of the World Wide Web, http://www.w3.org/TR/webarch/.

[XML-SCHEMA] Thompson, Henry S. et al., editors. XML Schema Part 1: Structure. W3C Recommendation 2 May 2001, http://www.w3.org/TR/xmlschema-1/.