

Title: Review of DCMI Application Profiles
Identifier: <http://gabriel.sub.uni-goettingen.de/~tbaker/profile-review/>
Main agenda: <http://gabriel.sub.uni-goettingen.de/~tbaker/madrid/>
Archived as: <http://dublincore.org/meetings/2005/09/profile-review/profile-review.pdf>
Modified: 2005-09-04 15:29, Sunday

One very important goal of the preliminary review of the application profile for Collection Description is to clarify the set of policies and guidelines used for the review of application profiles generally.

1. The review process

1.1 Discussion on review of profiles, May 2005

4 <http://gabriel.sub.uni-goettingen.de/~tbaker/madrid/files/2005-07-19.profile-review.txt>

To prepare for the review, start by re-reading very carefully these excerpts from the meeting notes, Washington, May 2005, summarizing our discussion about the review of profiles.

1.2 DCMI-compliant 'term' decision tree - draft

8 <http://www.ukoln.ac.uk/metadata/dcmi/term-decision-tree/>

This draft decision tree is about evaluating terms for compliance with the DCMI Abstract Model (see 3.3, not included in packet).

1.3 Dublin Core Application Profile Guidelines

11 <http://stage.dublincore.org/usage/documents/2005/09/03/profile-guidelines/>

Some changes are needed in this document:

- There is an action on Tom to edit these guidelines to strengthen the requirement of assigning URIs to any non-previously-existing terms. The draft "Guidelines for Assigning Identifiers to Metadata Terms" (see 3.4, not included in packet) could be reviewed for DCMI status and cited here in this regard.
- Diane has also raised some issues on the controlled vocabularies used to specify obligation (see 1.5, not included in packet).
- The document should perhaps summarize and cite the draft "XML, RDF, and DCAPs" (see 3.5, not included in this packet), which describes the differences between DC elements, XML elements, and RDF properties.
- The document should also summarize and point to "Element Refinement in Dublin Core Metadata" a DCMI Recommended Resource (see 3.6, not included in this packet).
- Other changes to the Guidelines may be necessary to bring this into line with the Abstract Model (e.g., to distinguish between Value Strings and Value URIs, as is done in the draft Collection Description profile).

1.4 DCMI Mixing and Matching FAQ - draft

24 <http://www.ukoln.ac.uk/metadata/dcmi/mixing-matching-faq/>

Another draft checklist, related to "XML, RDF, and DCAPs" (see 3.5, not included in this packet), which "attempts to answer some of the practical questions that implementers ask when faced with a desire to incorporate their existing XML metadata semantics into DCMI metadata applications".

1.5 Review of PBCore, June 2004

28 <http://gabriel.sub.uni-goettingen.de/~tbaker/madrid/files/2004-06-22.PBCore-final.txt>

This is the result of the first experience of the Usage Board with reviewing an application profile. The review was not as thorough as future reviews are currently intended to be. The review did not result in endorsement or the assignment of status -- only in feedback to the developers of PBCore.

1.5 Diane on controlled vocabularies specifying obligation

31 <http://gabriel.sub.uni-goettingen.de/~tbaker/madrid/files/2005-08-06.diane-on-dcap-guidelines.txt>.

2. Policies and processes

2.1 DCMI Usage Board Review of Application Profiles

32 <http://dublincore.org/usage/documents/profiles/>

This short document has not been edited in awhile but could be used as a placeholder for more extensive policies and guidelines, unless we were to conclude they should be documented elsewhere, e.g., in the Usage Board Administrative Process.

2.2 DCMI Usage Board Administrative Process

34 <http://gabriel.sub.uni-goettingen.de/~tbaker/madrid/files/2005-08-28.Process-for-profiles.pdf>

This is a four-page excerpt of the relevant parts of the draft Process document.

2.3 DCMI Namespace Policy

The current policy is:

38 <http://dublincore.org/documents/dcml-namespae/>.

A draft revision,

42 <http://www.ukoln.ac.uk/metadata/dcml/namespace-policy/>,

will be discussed in Madrid in the DCMI Architecture Working Group. Both versions of this important document have been included in this packet.

3 Other relevant documents

3.1 DCMI Policy on Naming Terms

<http://dublincore.org/documents/naming-policy/>

3.2 Procedure for Approval of DCMI Metadata Terms and Recommendations

<http://dublincore.org/usage/documents/approval/>

3.3 DCMI Abstract Model (DCMI Recommendation)

<http://dublincore.org/documents/abstract-model/>

3.4 Guidelines for Assigning Identifiers to Metadata Terms

<http://www.ukoln.ac.uk/metadata/dcml/term-identifier-guidelines/>

3.5 XML, RDF, and DCAPs

<http://www.ukoln.ac.uk/metadata/dcml/dc-elem-prop/>

3.6 Element Refinement in Dublin Core Metadata (draft DCMI Recommended Resource)

<http://dublincore.org/documents/dc-elem-refine/>

Topic: Usage Board meeting agenda, 9-10 September 2005
 Identifier: <http://gabriel.sub.uni-goettingen.de/~tbaker/madrid/>
 Archived as: <http://dublincore.org/meetings/2005/09/madrid/meeting-packet.pdf>
 Modified: 2005-09-04 15:29, Sunday

Attendance	Guests
Tom Baker	Joe Tennis
Andy Powell	Alistair Miles
Andrew Wilson	Pete Johnston
Akira Miyazawa	Dan Brickley
Diane Hillmann	
Stuart Sutton	

Friday morning

Pg Application profile review

- 2 01. Review of Collection Description application profile [Andrew]
<http://gabriel.sub.uni-goettingen.de/~tbaker/madrid/profile-collection/>
02. Review of Application Profiles [Tom]
 See separate meeting packet:
gabriel.sub.uni-goettingen.de/~tbaker/profile-review/

Friday afternoon

Pg Changes and guidelines for existing DCMI terms

- 3 03. Editorial changes to the DCMI Type Vocabulary [Stuart]
<http://gabriel.sub.uni-goettingen.de/~tbaker/madrid/type-vocabulary/>
- 4 04. Changes to DCMI properties [Andy]
<http://gabriel.sub.uni-goettingen.de/~tbaker/madrid/property-definitions/>
- 5 05. DC property domains and ranges [Andy]
<http://gabriel.sub.uni-goettingen.de/~tbaker/madrid/property-domains/>
- 6 06. Issues related to dc:date [Tom]
<http://gabriel.sub.uni-goettingen.de/~tbaker/madrid/date-issues/>

Saturday

Pg Policy and process

- 7 07. Revision of DCSV documents [Andy and Andrew]
<http://gabriel.sub.uni-goettingen.de/~tbaker/madrid/dcsv/>
- 8 08. MARC Relators - follow-up [Tom]
<http://gabriel.sub.uni-goettingen.de/~tbaker/madrid/marc-relators/>
- 10 09. Ongoing review of process document [Diane and Stuart]
<http://gabriel.sub.uni-goettingen.de/~tbaker/madrid/process/>
- 11 10. Guidelines for some new elements [Diane]
<http://gabriel.sub.uni-goettingen.de/~tbaker/madrid/new-term-guidelines/>
- 13 11. Accessibility decision - follow-up [Tom]
<http://gabriel.sub.uni-goettingen.de/~tbaker/madrid/accessibility/>
- 14 12. The back burner [Tom]
<http://gabriel.sub.uni-goettingen.de/~tbaker/madrid/backburner/>

2005-07-19: Review of Application Profiles - excerpts from
notes of May 19 (Washington) meeting

Agreed that general shift in focus from individual terms to application profiles is a good thing.

However, several questions/issues:

- * Need to clarify what "approving an application profile" means.
- * What does adding a term to an extension namespace mean -- I.e. what is DCMI saying by doing this?
- * What value is there in short term commitment to maintenance?
- * What is the current DCTERMS namespace for if we move to extension namespaces?
- * The bulk of UB work is in considering new terms - what will UB do less of once we have extension namespaces?
- * DCMI WGs have a requirement for creating and managing vocabularies -- how do extension namespaces help?
- * What do we want to say about application profiles and their component terms?
- * What is a "Strategic activity"?
- * Are there legal and social issues with DCMI assigning URIs to terms (particularly encoding schemes) that refer to things owned by other people?

Note that "conforming" currently means "conforms to grammatical principles" and the UB process (section 4.0) also says that conforming terms "do not conflict with or create ambiguity with existing terms" and that they are not both "cross domain" and "resource discovery" -- we have no current way of saying that a term simply "conforms to the AM". "Recommended" currently means "conforming" and "cross domain" and "resource discovery".

Could possibly revise the meanings of our statuses to be:

- * Conforming = conforms with AM
- * Recommended = conforming plus no overlaps with existing terms

Whether a term is "cross-domain" or not is orthogonal to this.

Need to define what "conforms to the AM" means -- need a decision tree. Action: Andy

Need to define other statuses (e.g. "Recommended") and what they mean -- need decision tree(s). Action: Diane, Stuart

These statuses and decision trees need to be able to be applied to individual elements, element refinements and encoding schemes and to application profiles.

What does a "review" of an application profile consist of?

- * Evaluate all elements, element refinements and encoding schemes for conformance with the AM

- o This should result in assigning a status of "conforming" or "recommended" to terms that currently don't have a status (new meanings - i.e. they all conform to the AM).
 - o Terms in non-DCMI namespaces should be documented according to DCMI requirements and have persistent URIs supported by known policies.
 - o All terms must have an assigned URI (by DCMI or by external organisation).
 - o If necessary, assign a URI in a DCMI namespace (DCTERMS or a DCMI extension namespace) and place documentation on the DCMI site.
- * Review AP-specific comments for clarity and to ensure that they do not extend existing term semantics
 - * Check that AP documentation conforms to DCMI requirements (CEN guidelines or revised version)
 - * Review and document evidence of community buy-in to the AP (as a whole)
 - * Consider whether the AP does what it says it does -- i.e. does it meet its documented functional requirements
 - * Produce a "review report"

If the AP meets all these requirements then it is "conforming".

Possible changes to CEN guidelines for AP -- Action: Tom

- * Details about assigning URIs to terms
- * Contextual info about why particular terms were chosen.

Process prior to UB meeting:

- * Assign shepherds (reports for packet by 1 Sept)
 - o Overall shepherd: Andrew
 - o Evaluation individual terms against AM: Andrew, Andy
 - o Check comments: Akira
 - o Check documentation: Tom
 - o Check community buy-in: Andrew
 - o Check functional requirements: Tom
- * There should be an single announcement about the AP and all terms within the AP that don not currently have a DCMI status on dc-general by the shepherd -- the comment period should be one month (announced Mon 25 July)
- * With regard to the AP we're looking for views on fitness for purpose, wider community buy-in and commentary on semantic overlaps -- primary measure of community buy-in is thru the WG.

Agreed that initial review of new terms and APs is to determine conformance to AM or not. Review of AP will also result in a "review report" -- but need to determine the level of comment in the report

Need longer term roadmap of where APs can go, e.g.

WG ----> conformance ----> WG -----> recommended.

Need documentation for minimal proposal of new term -- Action:

Suggestion that

What namespace do "new" terms go into?

OPTION 1

- * If elements, element refinements that are not in other namespaces are "recommended" and "cross-domain" then they go into the DCTerms namespace. Otherwise they go into a DCMI extension namespace.
- * All vocabulary terms that are not in other namespaces go into a DCMI extension namespace.

OPTION 2

- * All "new" elements, element refinements and encoding schemes that are not in other namespaces go into DCTERMS namespace.
- * All vocabulary terms that are not in other namespaces go into a DCMI extension namespace.

OPTION 3

- * Everything goes into an extension namespace

OPTION 4 (as per BoT proposal)

- * If proposal originates from a strategic activity it goes into an extension namespace
- * Everything else goes into DCTerms.

What namespace URI should we use?

Depends on the OPTIONS above.

If OPTION 1, proposal to use:

<http://purl.org/dcx/cdwg/> - an acronym with implied semantics
 OR <http://purl.org/dcx/2005/06/> - a date stamp
 OR <http://purl.org/dcx/01/> - something neutral (eg, sequential number)

If OPTION 2, proposal to use:

http://purl.org/dcx/vocab_name/

Summary position of UB on extension namespaces proposal:

- * Agree with move to using APs as primary mechanism for proposing new terms. Will develop documentation for WGs to encourage and support this.
- * Agree with need to provide home for WG-generated vocabularies. Will develop new namespaces and associated documentation for vocabularies.
- * It is not clear that there is currently a demonstrable need for new namespaces for elements, element refinements and encoding schemes, and we see no pressing need to put new terms into extension namespaces. Under proposals above the "approval" process for new elements, element refinements and encoding schemes is unchanged -- therefore can continue using existing DCTERMS namespace for them, at least until we see how

things progress as a result of moving to greater use of APs.

- * Recognise a need to structure documentation about DCMI terms to emphasise a core set and to de-emphasise domain-specific terms. The UB proposes to address this problem by reviewing the statuses we assign to terms and the ones already assigned.

DCMI-compliant 'term' decision tree

Andy Powell
UKOLN, University of Bath
July 2005

This decision tree can be used to see if something is a DCMI-compliant *element*, *element refinement* or *encoding scheme*, where "DCMI-compliant" means conformant with the [DCMI Abstract Model](#) and therefore suitable for use in DC metadata descriptions.

Note that in the following text, the italicised terms are defined in the terminology section below.

Decision tree

1. Has the thing been explicitly declared as a DCMI *element* (i.e. as a *property*)?

The declaration may take the form of a human-readable statement, e.g.

X is an 'element' as defined in the DCMI Abstract Model

or as a machine-readable RDFS declaration

```
<rdf:Property rdf:about="http://example.org/term/X">
  ...
</rdf:Property>
```

- o If 'yes', go to question 2.
- o Otherwise, go to question 3.

2. Are the expected values of the *element* resources that have been assigned 'value URIs' or that can be represented using simple 'value strings' (plain text strings)?

- o If 'yes', go to question 9.
- o Otherwise, go to question 3.

3. Has the thing been explicitly declared as a DCMI *element refinement* (i.e. as a *property*)?

The declaration may take the form of a human-readable statement, e.g.

X is an 'element refinement' as defined in the DCMI Abstract Model

or as a machine-readable RDFS declaration

```
<rdf:Property rdf:about="http://example.org/term/X">
  ...
</rdf:Property>
```

- o If 'yes', go to question 4.
- o Otherwise, go to question 5.

4. Are the expected values of the *element refinement* resources that have been assigned 'value URIs' or that can be represented using simple 'value strings' (plain text strings)?

- o If 'yes', go to question 9.
- o Otherwise, go to question 5.

5. Has the thing been explicitly declared as a DCMI *syntax encoding scheme*?

The declaration may take the form of a human-readable statement, e.g.

X is a 'syntax encoding scheme' as defined in the DCMI Abstract Model

or as a machine-readable RDFS declaration

```
<rdf:Description rdf:about="http://example.org/term/X">
  <dc:type rdf:resource="http://dublincore.org/usage/documents/principles/#syntax-encoding-scheme"/>
  ...
</rdf:Description>
```

- o If 'yes', go to question 6.
 - o Otherwise, go to question 7.
6. **Are all the valid values according to the *syntax encoding scheme* simple 'value strings' (plain text strings)?**
- o If 'yes', go to question 9.
 - o Otherwise, go to question 7.
7. **Has the thing been explicitly declared as a DCMI *vocabulary encoding scheme*?**
The declaration may take the form of a human-readable statement, e.g.

X is a 'vocabulary encoding scheme' as defined in the DCMI Abstract Model

or as a machine-readable RDFS declaration

```
<rdfs:Class rdf:about="http://example.org/term/X">
  ...
</rdfs:Class>
```

- o If 'yes', go to question 9.
 - o Otherwise, the thing is not a valid DCMI *element*, *element refinement* or *encoding scheme*.
8. **Are all the valid members of the *vocabulary encoding scheme* resources that have been assigned 'value URIs' or that can be represented using simple 'value strings' (plain text strings)?**
- o If 'yes', go to question 9.
 - o Otherwise, the thing is not a valid DCMI *element*, *element refinement* or *encoding scheme*.
9. **Has the thing been assigned a URI (a *property URI* or an *encoding scheme URI*)?**
For example: <http://example.org/term/X>
- o If 'yes', the thing is a valid DCMI *element*, *element refinement* or *encoding scheme*.
 - o Otherwise, the thing is not a valid DCMI *element*, *element refinement* or *encoding scheme*.

Note

The exact details of the RDF declarations about are not fully agreed. The examples used here are based on the current DCMI RDF declarations, with a little modification. The author suggests that the 'term type' URIs used here (e.g. <http://dublincore.org/usage/documents/principles/#vocabulary-encoding-scheme>) need to be changed.

Terminology

class

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

class URI

A URI that identifies a class.

element

A property of a resource.

element refinement

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

A vocabulary encoding scheme or a syntax encoding scheme.

encoding scheme URI

A vocabulary encoding scheme URI or a syntax encoding scheme URI.

property

A specific aspect, characteristic, attribute, or relation used to describe resources.

property URI

A URI that identifies a single property.

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

term

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

A URI that identifies a term.

vocabulary encoding scheme

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

Page maintained by: Andy Powell

Last updated: 21-Jul-2005



[Home](#) > [Usage](#) > [Documents](#) > [2005](#) > [09](#) > [03](#) > [Profile-guidelines](#) >

Dublin Core Application Profile Guidelines

Creator: Thomas Baker

Creator: Makx Dekkers

Creator: Thomas Fischer

Creator: Rachel Heery

Date issued: 2005-09-03

Identifier: <http://dublincore.org/documents/2005/09/04/profile-guidelines/>

Replaces: Not applicable

Is Replaced By: Not applicable

Latest Version: <http://dublincore.org/documents/profile-guidelines/>

Status of Document: This is a DCMI Working Draft.

Description: These guidelines specify the structure and content of Dublin Core Application Profiles, a form for documenting which terms a given application uses in its metadata, with what extensions or adaptations, and specifying how those terms relate both to formal standards such as Dublin Core as well as to less formally defined element sets and vocabularies. The document on which this is based was originally developed in the context of the CEN/ISSS Workshop on Metadata for Multimedia Information - Dublin Core (WS/MMI-DC) of CEN, the European Committee for Standardization and was published in 2003 as the CEN Working Agreement CWA 14855.

The text of this version is substantially identical (minus introductory text related to CEN procedure and a table of contents) to the text of CWA 14855, from which future revisions will increasingly diverge.

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 distills 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 documentary 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	<p>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/.</p> <p>All Dublin Core terms are fully documented at http://www.dublincore.org/documents/dcml-terms/.</p>

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 18 th century, put '18 th 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/>.



Metadata associated with this resource: <http://dublincore.org/usage/documents/2005/09/03/profile-guidelines/index.shtml.rdf>

[Copyright](#) © 1995-2005 [DCMI](#) All Rights Reserved. DCMI [liability](#), [trademark/service mark](#), [document use](#) and [software licensing](#) rules apply. Your interactions with this site are in accordance with our [privacy](#) statements. Please feel free to [contact us](#) for any questions, comments or media inquiries.

DCMI and the DCMI Web site are hosted by [OCLC Research](#).

DCMI Mixing and Matching FAQ

Andy Powell
UKOLN, University of Bath

This document attempts to answer some of the practical questions that implementors ask when faced with a desire to incorporate their existing XML metadata semantics into DCMI metadata applications.

Question 1: My favorite XML schema contains an element type or attribute (e.g. `my:price` or `my:currency`) that I want to use in my Dublin Core metadata. How do I do it?

The bad news is that an existing XML element type or attribute **cannot** be used as is in DCMI metadata applications. This is a very important point, but is sometimes hard for people to understand. Before you can use your favorite element or attribute you must declare it as a DCMI-compatible term. The good news is that doing so need not be an overly onerous task. Here's what you have to do:

1. Decide whether your XML element type or attribute corresponds to DCMI's notion of a 'property' or an 'encoding scheme'. These notions are defined in the [DCMI Abstract Model](#) but, for convenience, the definitions are repeated here:

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

Encoding scheme is the generic name for vocabulary encoding scheme and 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.

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.

2. Next, check whether an equivalent property or encoding scheme has already been defined by the DCMI (or elsewhere), for use in DCMI metadata. A good place to start checking is the list of [DCMI Metadata Terms](#).
3. Next, assign a URI reference to your new property or encoding scheme - see question 3 below.
4. Finally, declare your new property or encoding scheme using the RDF Schema language (RDFS) and make this declaration available somewhere on the Web - see questions 4 and 5 below.

Question 2: Why can't I just re-use my XML element type or attribute as is?

Because XML element types and XML attributes are component parts of an XML language. Their meaning is determined solely by their placement in the XML tree structure of the given XML language and the semantics that the developers of that language chose to associate with that structure. On the other hand, DCMI properties and encoding schemes are conceptual entities within the [DCMI Abstract Model](#) - their meanings are defined by the model and by the semantic declarations that DCMI make available. Furthermore, XML element types and attributes are named using XML *expanded names* (a pair comprising an *XML Namespace Name* (which is a URI reference) and a *local name*). On the other hand, DCMI properties and encoding schemes are named using URI references.

So, although XML element types and DCMI properties may look superficially similar, for example `lom:title` looks similar to `dc:title` when the two are encoded in XML, in fact they are very different entities.

For those who are interested, the [XML, RDF and DCAPs](#) document provides a much more in-depth treatment of the differences

between XML element types and RDF properties and the usage of both in the context of DCMI metadata.

Question 3: How do I assign a URI to my 'element'?

Unfortunately, there is little best-practice in this area to draw on at the time of writing. The [Guidelines for assigning identifiers to metadata terms](#) document lists some possible approaches.

One immediate issue to consider is whether to make the URI reference that is assigned to the new property or encoding scheme similar to the XML expanded name of the XML element or attribute. DCMI has chosen to keep the two things very similar. For example, the XML expanded name that is used to represent the DC Title property according to the [Guidelines for encoding DC in XML](#) recommendation is `dc:title` (corresponding to the `http://purl.org/dc/elements/1.1/` XML namespace name and the `title` local name). The DC Title property is assigned the `http://purl.org/dc/elements/1.1/title` URI reference. Therefore, the property URI reference is simply a concatenation of the component parts of the XML expanded name.

On the other hand, the RDF encoding of the IEEE LOM (which has effectively made the LOM available for use with DCMI metadata because the DCMI Abstract Model is essentially the same as the RDF model) has chosen to keep the XML expanded names used in the LOM/XML encoding and the URI references assigned to the LOM/RDF properties completely separate.

Example to be provided

Remember that your new property is likely to appear in the various DCMI encoding syntaxes using the name that is the final part of the URI reference you assign (usually the bit after the final '/' or '#'). For example, the URI reference `http://example.com/my/terms#color` is likely to appear as `my:color` (in XML syntaxes) or `MY.color` (in HTML syntaxes).

If in doubt, register with the [PURL](#) system and assign a PURL to your new property or encoding scheme.

Question 4: How do I declare my XML element type or attribute as a DCMI property?

Use the RDF Schema language to do this. Take a look at the [DCMI RDFS terms declarations](#) for inspiration! As a minimum, you'll need something like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE rdf:RDF>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:dcterms="http://purl.org/dc/terms/"
  xmlns:dc="http://purl.org/dc/elements/1.1/"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">

  <rdf:Property rdf:about="http://purl.org/my/terms/price">
    <rdfs:label xml:lang="en-US">Price</rdfs:label>
    <rdfs:comment xml:lang="en-US">The amount of money needed to purchase the resource.
  </rdfs:comment>
    <rdfs:isDefinedBy rdf:resource="http://purl.org/my/terms/" />
    <dcterms:issued>2004-12-03</dcterms:issued>
    <dcterms:modified>2005-02-21</dcterms:modified>
    <dc:type rdf:resource="http://dublincore.org/usage/documents/principles/#element" />
  </rdf:Property>
</rdf:RDF>
```

Make sure that this RDF/XML document is available (in this case) at both `http://purl.org/dc/terms/` and `http://purl.org/my/terms/price`.

Question 5: How do I declare my XML element type or attribute as a DCMI encoding scheme?

Use the RDF Schema language to do this. Take a look at the [DCMI RDFS terms declarations](#) for inspiration! As a minimum, you'll need something like this:

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE rdf:RDF>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:dcterms="http://purl.org/dc/terms/"
  xmlns:dc="http://purl.org/dc/elements/1.1/"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">

  <rdfs:Class rdf:about="http://purl.org/my/terms/PoundsSterling">
    <rdfs:label xml:lang="en-US">Pounds Sterling</rdfs:label>
    <rdfs:comment xml:lang="en-US">A price in UK pounds sterling, formatted in the
following way: "P.pp"
    (where "P" represents one or more digits for the number of pounds and "pp"
represents
    two digits for the number of pence).</rdfs:comment>
    <rdfs:isDefinedBy rdf:resource="http://purl.org/dc/terms/" />
    <dcterms:issued>2003-07-11</dcterms:issued>
    <dcterms:modified>2004-06-15</dcterms:modified>
    <dc:type rdf:resource="http://dublincore.org/usage/documents/principles/#encoding-
scheme" />
  </rdfs:Class>
</rdf:RDF>

```

The examples used in questions 4 and 5 would allow the following XML fragment to be used in a DC/XML document that conforms to the [Guidelines for implementing Dublin Core in XML](#) recommendation:

```
<my:price xsi:type="my:PoundsSterling">2.99</my:price>
```

assuming that the appropriate namespace declarations are in place. Note that, by convention, the XML *local name* for an encoding scheme starts with an uppercase letter.

Question 6: I still don't understand! Do you have another example?

OK. Let's say that I want to start using DC metadata to describe car parts, and that my company (ZZ Motors) already uses an XML schema that allows for XML fragments like this:

```

<zz:carpart>
  <zz:engine>
    <zz:type>petrol</zz:type>
    <zz:capacity>2000cc</zz:capacity>
  </zz:engine>
  <zz:fueltank>
    <zz:capacity>25l</zz:capacity>
  </zz:fueltank>
</zz:carpart>

```

'zz' being associated with the `http://zz.com/carparts/` XML namespace name.

For the sake of argument, let's say that I want to start using a property in my DC metadata that indicates the engine capacity. Looking at my existing XML schema, I note that I already have an XML element type with the local name `capacity` (under `zz:engine`) that I might be able to use? But there's a problem! I also have an XML element type with the local name `capacity` elsewhere in my XML tree structure (under `zz:fueltank`). So I cannot simply use 'capacity' as the local name when I'm thinking about assigning a URI reference to my new property.

The semantics of my current XML element types called `capacity` are determined by the placement of those two element types in the XML tree. In fact I have two 'properties', which we'll call `engineCapacity` and `fueltankCapacity`. I'm interested in the first of these.

OK, so now I need to assign a URI reference to my new property called `engineCapacity`. I want this property to be widely used (because it'll make my supply chains work more smoothly if everyone else uses the same property) so I decide to name my

new property using a PURL, rather than a URI reference somewhere under my company's DNS domain name. I choose:

```
http://purl.org/carparts/terms/engineCapacity
```

Now I need to declare my new property using RDFS. I create a file on my company's Web site that contains the following:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE rdf:RDF>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:dcterms="http://purl.org/dc/terms/"
  xmlns:dc="http://purl.org/dc/elements/1.1/"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">

  <rdf:Property rdf:about="http://purl.org/carparts/terms/engineCapacity">
    <rdfs:label xml:lang="en-US">Engine Capacity</rdfs:label>
    <rdfs:comment xml:lang="en-US">The total combustion chamber size of an engine in
cubic centimetres.</rdfs:comment>
    <rdfs:isDefinedBy rdf:resource="http://purl.org/my/terms/" />
    <dcterms:issued>2005-02-21</dcterms:issued>
    <dcterms:modified>2005-02-21</dcterms:modified>
    <dc:type rdf:resource="http://dublincore.org/usage/documents/principles/#element" />
  </rdf:Property>
</rdf:RDF>
```

Finally, I register two PURLs, <http://purl.org/carparts/terms/> and <http://purl.org/carparts/terms/engineCapacity>, and configure them both so that they resolve to the RDF/XML document (above) on my Web server.

Content by: Andy Powell
Last updated: 22-Feb-2005

Title: Comments on PB Core
Creator: DCMI Usage Board
Date: 2004-06-22

At its meeting in Bath, UK in March 2004, the DCMI Usage Board discussed the metadata element set "PB Core", which was developed for the public broadcasting community on the basis of Dublin Core metadata elements [1]. The Usage Board considers application profiles in terms of their conformance with Dublin Core principles [2]. The following is not a thorough review of PB Core but an overview of aspects which are problematic from the standpoint of an application profile conforming to DCMI principles.

There are several areas in which PB Core diverges from established DCMI principles and practices:

1. Use of the "dot" syntax to form metadata element names. This naming style was used in HTML implementations of Dublin Core of the late 1990s but has been superseded by a naming style (and data model) which considers both elements and element refinements as "properties" -- and accordingly with names that stand on their own (e.g., see Footnote 9 of [3], the "DCMI Policy on Naming Terms" [4], and the section on Element Refinements in [5]). In PB Core, then "Title.Program" would be "programTitle".
2. Element Refinements must refine -- but not extend -- the semantics of an element (see [2]). Some of the element refinements proposed in PB Core, however, change or broaden the meaning of the base element. Examples are:

Title.Series

The semantics of dc:title are: "Definition: A name given to the resource. Comment: Typically, Title will be a name by which the resource is formally known." "The resource" means here: "the resource being described by dc:title".

However, pbcore:Title.Series does not refer to "the resource being described by dc:title". Rather, it refers to the title of the series to which the resource being described by dc:title belongs.

In this case, the Series may be seen as a separate entity -- it is in effect a "related resource" (see the draft Abstract Model [6] for a discussion of "related resource"). The relationship between a described resource and a related resource may be described with dc:relation. The DC Libraries Working Group discussed such a case in the context of the DC-Lib Application Profile and recommended pointing to related resources such as Series with the term dct:isPartOf, a refinement of dc:relation [7]. (In principle, one might coin an even narrower term such as pbs:isPartOfSeries, but such a local refinement would presumably not be understood outside of PBS.)

Language.Usage

The semantics of dc:language are: "Definition: A language of the intellectual content of the resource. Comment: Recommended best practice is to use RFC 3066 [RFC3066] which, in conjunction with ISO639 [ISO639]), defines two- and three-letter primary language tags with optional subtags. Examples include "en" or

"eng" for English, "akk" for Akkadian", and "en-GB" for English used in the United Kingdom."

In the draft PB Core, the intended use of Language.Usage is to record the existence and type of other audio and textual representations of the main audio or language presentation mode for a resource or asset. In the draft PB Core, suggested values for the element include controlled terms such as "DVD Subtitle01".

However, while "eng" ("English") is a proper value of dc:language, "DVD Subtitle01" is not. Rather, the PB Core drafters mean to say, in effect: "This resource (a DVD) also has subtitles, which have the language English."

In modeling terms, one might either describe each set of subtitles or captions as a "related resource" with attributes of its own (such as dc:language); this solution would be formally precise but complex and expensive to implement. Alternatively, one might more simply see the languages of these multiple component parts all as attributes of the DVD as a whole. The metadata would then say, in effect, "This DVD has the languages English, Spanish, and Portuguese" without distinguishing between subtitles and captions.

These two options involve trade-offs between simplicity (which in this case would be simplistic) and complexity (which would be expensive), with implications for interoperability. One compromise solution would be to create PBS-specific refinements such as "Subtitle Language" or "Caption Language", each of which would hold "a language of the resource" (i.e., a language code) and thus dumb down properly to Language.

Applications outside of PBS -- unaware perhaps of these distinctions -- might interpret these different refinements in an undifferentiated way as meaning "This DVD has the languages English, Spanish, and Portuguese" (see above). In other words, the distinctions between subtitles and captions might be lost to non-PBS users of PBS metadata. This approach could work, however, if it is sufficient for PBS that the distinctions be understood internally.

3. Some PB Core "element refinements" are not element refinements in the sense of the DCMI model. Examples are:

Relation.Type
Creator.Role

Both cases in effect imply a data model in which a Dublin Core element -- e.g., dc:relation or dc:creator -- itself has an attribute which, in turn, has a value in which controlled vocabularies (e.g., "IsPartOf" for Relation, "Cinematographer" for Creator) are used.

In the DCMI model [1,5], however, terms such as IsPartOf or Cinematographer are themselves "element refinements" with respect to elements. In other words, the term identified by the URI <http://purl.org/dc/terms/isPartOf> semantically refines the term <http://purl.org/dc/elements/1.1/relation>.

4. Some element refinements violate the One-to-One Principle, according to which Dublin Core metadata describes (ideally) one manifestation or version of a resource, rather than conflating multiple manifestations into one description

(see Usage Guide [7], Section 1.2.). An example of this is:

Description.ProgramRelatedText

This is described as the actual text or link to text. This violates the One-to-One Principle inasmuch it is really another representation of the program in textual form -- a "related resource" and, as such, better referenced using a refinement of dc:relation.

5. Historically, the Dublin Core community has had difficulty distinguishing within the dc:type element between "form" and "genre". Rather than create two separate refinements of dc:type -- Type.Form and Type.Genre, as proposed in the draft PB Core -- it may be easiest simply to use two or more specialized vocabularies (one vocabulary of "forms" and one vocabulary of "genres") in conjunction with the unrefined element dc:type.

For presenting PB Core as a document, its authors might consider using the CEN guidelines for Dublin Core application profiles [9].

- [1] <http://dublincore.org/usage/meetings/2004/03/ISSUES/profiles-pbcore/>
- [2] <http://dublincore.org/usage/documents/principles/>
- [3] <http://dublincore.org/usage/documents/2003/11/18/principles/>
- [4] <http://dublincore.org/documents/naming/>
- [5] <http://dublincore.org/documents/dcml-terms/>
- [6] <http://www.ukoln.ac.uk/metadata/dcml/abstract-model/2004-02-04/>
- [7] <http://dublincore.org/documents/2002/09/24/library-application-profile/>
- [8] <http://dublincore.org/documents/usageguide/>
- [9] <http://www.cenorm.be/iss/cwal4855/>

Date: Sat, 6 Aug 2005 20:25:42 -0400 (EDT)
Subject: AP Question
From: "Diane Ileana Hillmann" <dihl@cornell.edu>
To: "Tom Baker" <baker@sub.uni-goettingen.de>

I'd like to bring a couple of AP issues to your attention, based on recent discussions with the DC-Ed AP Drafting Committee as well as a result of my just-ended workshop on APs in Canada. I know you're involved with the revision of the CEN Guidelines and I trust you will forward these comments on to others interested in these issues.

The issue has to do with the terms used to specify obligation in an AP. Like many developing APs, we've copied much from the Libraries AP, including their terms and codes for obligation. After coming back from my vacation in France (with considerably fewer brain cells than I began with) I looked at our AP and started seeing "R" as Required (neglecting to look at what we'd copied over as a key) and the confusion from that caused me to look a bit more closely at what we'd cut and pasted. It became clear to me that the DC-Lib group had imported those terms and codes from MARC, and there were two issues that started to jell:

1. There seems no other places in the AP that codes are used, and given my own brain fog it started to seem like a bad idea to use codes rather than terms. For catalog librarians these codes are second nature; for others, not so.
2. Once I started thinking about the terms, it became obvious that what we were dealing with here was a small controlled vocabulary, with terms that we hadn't given much thought to as we incorporated them.

I started asking myself whether this was the right set of terms to begin with, and whether it might make sense to ask you about whether any notion of standardizing obligation terms might be part of the work being attempted as the Guidelines are reviewed.

Now, having spent three days talking about APs (and nothing but APs) with a small group of very savvy and engaged folks, I'm more and more convinced that this issue is an important one. For instance, one of the participants wanted to include in his AP terms to describe:

- A. Information that might be carried if received from others but not necessarily sought prospectively
- B. Information that will not be used, neither stored nor accepted (particularly if it might be pernicious or misleading in a particular context).

Maybe because it was the end of a long day, but I couldn't figure out why not, and this lead me to think that the issue might benefit from further discussion.

I'd appreciate any insights you might offer on these issues, particularly the one concerning codes vs. terms, as we'd like to get that one straight before moving ahead (and it might be an easier one, on the whole!)



[Home](#) > [Usage](#) > [Documents](#) > [Profiles](#) >

Title: DCMI Usage Board Review of Application Profiles
 Creator: Thomas Baker
 Identifier: <http://dublincore.org/usage/documents/2003/02/11/profiles/>
 Latest version: <http://dublincore.org/usage/documents/profiles/>
 Date modified: 2003-02-11
 Description: This document defines the term "Application Profile" in the context of the Dublin Core Metadata Initiative. Criteria for Usage Board review of Application Profiles and guidelines for submission are outlined in the DCMI Usage Board Administrative Processes document [PROCESS].

"Application Profile" defined

For the purposes of DCMI Usage Board review, an Application Profile (AP) is a declaration of which metadata terms an organization, information resource, application, or user community uses in its metadata. Moreover:

- By definition, an AP cannot "declare" new metadata terms and definitions; it only "reuses" terms from existing element sets [HEERY].
- The ideal element set will use URIs to uniquely identify its terms within XML namespaces [DCMI-NAMESPACE]. As of 2002, however, this cannot be required.
- By definition, any new term coined for use in an AP must first be declared in a form citable in the AP.
- An AP may also provide additional documentation on how the terms used are constrained, encoded, or interpreted for particular purposes.

As of 2002, APs are seen primarily as a form of documentation, the purpose of which is to help implementor communities harmonize their metadata practice. It is hoped that in the longer term, machine-processable versions of such APs based on data models such as RDF will provide a basis for automating metadata interoperability functions such as semantic crosswalks and format conversions.

References

[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/>.

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

[PROCESS] <http://dublincore.org/usage/documents/process/>.



Metadata associated with this resource: <http://dublincore.org/usage/documents/profiles/index.shtml.rdf>

[Copyright](#) © 1995-2005 [DCMI](#) All Rights Reserved. DCMI [liability](#), [trademark/service mark](#), [document use](#) and [software licensing](#) rules apply. Your interactions with this site are in accordance with our [privacy](#) statements. Please feel free to [contact us](#) for any questions, comments or media inquiries.

DCMI and the DCMI Web site are hosted by [OCLC Research](#).

documentation by the UB Chair.

1.2. Proposals for terms accompanying application profile submissions

1.2.1. New terms appearing in application profile submissions must be evaluated for compliance with the DC Abstract Model prior to evaluation of the Application Profile itself.

1.2.2. New terms deemed in compliance with the DC AM may be registered in DC-hosted namespaces as 'conforming'

1.3. Proposals for endorsement of terms ~~{(e.g., LOC relators)}~~ in other namespaces for use within Application Profiles

1.3.1. Existing terms housed in other namespaces to be used within Application Profiles seeking review must be evaluated for compliance with the DC Abstract Model prior to evaluation of the Application Profile itself.

1.3.2. Existing terms deemed in compliance with the DC AM may be noted as 'endorsed' within the registered Application Profile.

1.4. Proposals for registration of application profiles

1.4.1. Sources of proposals

1.4.1.1. Existing working groups or working groups established for the purpose of developing proposals

1.4.1.1.1. Designation of application profile as DCMI 'strategic activity'

1.4.1.2. Metadata implementers (established projects, communities or research groups)

1.4.1.3. UB itself

1.4.2. ~~For the purposes of review~~ Review by the Usage Board:

1.4.2.1. The Usage Board is interested in reviewing application profiles that make substantial use of Dublin Core elements. The review of application profiles by the Usage Board serves to:

1.4.2.1.1. analyze the usage of Dublin Core within significant implementations;

1.4.2.1.2. assign a DCMI status of "stamp of approval conforming;"

1.4.2.1.3. promote the sharing of application profiles between

communities; and

1.4.2.1.4. identify new terms as candidates for inclusion in **DCMI-hosted** namespaces.

1.4.2.2. Application profiles must provide, for each term, an identifier of the element set where it is defined, ideally in the form of URIs for individual terms.

1.4.2.3. If the terms in an application profile describe anything other than generic "resources" (the typical domain of Dublin Core), the application profile must make this clear. This is particularly important if an application profile is based on a data model that describes multiple classes of resources, such as agents or collections.

1.4.2.4. It is recommended that application profiles be prepared using the Dublin Core Application Profile guidelines published by CEN [<http://www.cenorm.be/cenorm/businessdomains/businessdomains/iss/cwa/cwa14855.asp>].

1.4.2.5. Each application profile must provide, or point to, a short text that describes:

1.4.2.5.1. The context and purposes in which the application profile is used or is likely to be used.

1.4.2.5.2. The organizations or individuals involved in its development and a capsule history thereof.

1.4.2.5.3. Any arrangements, policies, or intentions regarding the future development and maintenance of the application profile.

1.4.3. [Results of](#) review of Application Profiles by the Usage Board

1.4.3.1. An application profile is "well-formed" if it is presented in accordance with the broad and flexible requirements outlined above. These presentation requirements may become more specific as "good practice" emerges over time.

1.4.3.2. **The Usage Board will evaluate terms to determine their conformance with the DCMI Abstract Model.**

1.4.3.3. The use of terms related to Dublin Core (such as refinements of Dublin Core elements, or Dublin Core elements that have been constrained for particular contexts) will be evaluated from the standpoint of semantic conformance, grammatical principle (eg, "dumb-down"), clarity, and good practice.

1.4.4. Publication and use of Usage Board Reviews

1.4.4.1. An application profiles that "pass" review will be assigned the status of 'conforming'.

1.4.4.1.1. The status of 'conforming' indicates a Usage Board assessment of the application profile as of the date of its submission for review.

1.4.4.1.2. Changes to already 'conforming' application profiles require further Usage Board review of the application profile in whole or in part according to the processes and criteria outlined in sections 6.1 through 6.3.

1.4.4.2. For application profiles that "pass" review, the Usage Board will publish a Review on a Web page for application profiles.

1.4.4.3. Each Review will include, at a minimum:

1.4.4.3.1. Any comments from the Usage Board on the application profile.

1.4.4.3.2. Pointers to locally archived copies of the application profile as originally submitted and (if necessary) as subsequently amended in light of Usage Board comments.

1.4.4.3.3. A pointer to the "latest version" of an application profile held by its maintainers.

1.4.5. Review represents a form of recognition, and its URL will be persistent for purposes of citation.

1.4.6. Registration of 'conforming' application profiles

1.5. Proposals for term revisions

1.5.1. DCMI namespace terms

1.5.1.1. General process for term changes

1.5.1.1.1. Requests to change terms in this namespace may originate within the Usage Board or externally.

1.5.1.1.2. A Usage Board member will be assigned to draft a proposal for a change

1.5.1.1.3. Changes provisionally approved by the Usage Board will be circulated for general comment on the DC-General discussion list for one month before final approval.

1.5.1.1.4. Final approval for term changes without significant opposition may be approved by email or teleconference vote.

1.5.1.2. Terms from namespace: <http://purl.org/elements/1.1/> require changes to the relevant standards: ISO Standard 15836-2003 (February 2003) and NISO

Standard Z39.85-2001 (September 2001)

1.5.1.3. Terms from DCMI hosted namespaces (to be added)

1.5.2. Application profile terms

1.5.2.1. Application profile terms residing on DCMI hosted namespaces will be subject to the same changes processes as other DCMI terms, but managed by the entities responsible for the terms.

1.5.2.2. Application profile terms residing on non-DCMI namespaces will be subject to term policies of the host entity.

1.6. Proposals for application profile revisions

1.6.1. Changes to already 'conforming' application profiles require further Usage Board review of the application profile in whole or in part according to the processes and criteria outlined in sections @@@ and @@@

1.6.2. Changes to DCMI-registered 'conforming' application profiles will be versioned according to DCMI namespace policies.

Part III—Usage Board: Internal Processes

1. Changes to Usage Board Procedures



[Home](#) > [Documents](#) > [Dcml-namespace](#) >

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/dcml-namespace/>

Replaces: <http://dublincore.org/documents/2001/09/17/dcml-namespace/>

Is Replaced By: Not Applicable

Latest version: <http://dublincore.org/documents/dcml-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:

<code>http://purl.org/dc/elements/1.1/</code>	Dublin Core Metadata Element Set, Version 1.1 (15 elements)
<code>http://purl.org/dc/terms/</code>	DCMI elements and DCMI qualifiers (other than those elements defined in the Dublin Core Metadata Element Set, Version 1.1 above)
<code>http://purl.org/dc/dcmitype/</code>	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>

Copyright © 1995-2005 DCMI All Rights Reserved. DCMI [liability](#), [trademark/service mark](#), [document use](#) and [software licensing](#) rules apply. Your interactions with this site are in accordance with our [privacy](#) statements. Please feel free to [contact us](#) for any questions, comments or media inquiries.

DCMI and the DCMI Web site are hosted by [OCLC Research](#).

Dublin Core Metadata Initiative logo	About the Initiative	Documents	Working Groups	Resources
	Dublin Core Metadata Initiative	Tools and Software	Projects	AskDCMI
Dublin Core Metadata Initiative				

[Home](#) > [Documents](#) > [Dcml-namespace](#) >

DRAFT UPDATE to 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](#)

Contributor: [Pete Johnston](#)

Date Issued: 2005-08-17

Identifier:

Replaces:

Is Replaced By: Not Applicable

Latest version:

Status of document: DRAFT UPDATE to DCMI Recommendation

Description of document: All *terms* used in metadata descriptions that conform to the DCMI Abstract Model [DCAM] must be assigned a unique URI [RFC3986]. For convenience, the *term URIs* that are assigned and managed by the DCMI are grouped into collections known as *DCMI namespaces*. This document describes how *term URIs* are allocated by the DCMI and the policies associated with *DCMI namespaces*.

Glossary:

This document uses the following terminology:

term

A property (i.e. element or element refinement), vocabulary encoding scheme, syntax encoding scheme or concept taken from a controlled vocabulary (concept space).

DCMI term

A *term* that is declared and maintained by the DCMI.

term URI

The URI that identifies a *term*.

term name

A human-readable token assigned to a *term*.

DCMI term URI

The URI for a *term* that is declared and managed by the DCMI.

vocabulary

A collection of *terms* (often as used in the context of an 'application profile').

DCMI namespace

A collection of *DCMI term URIs* where each *term* is assigned a URI that starts with the same 'base URI'. The 'base URI' is known as the *DCMI namespace URI*. (Note that a *DCMI namespace* is not the same as an 'XML namespace').

DCMI namespace URI

The URI that identifies a *DCMI namespace*.

DCMI recommendation

A human-readable document that may define one or more *DCMI terms*.

DCMI term declaration

A machine-processable representation of one or more *DCMI terms*, expressed in a schema language.

Note that the grouping of *term URIs* into a *DCMI namespace* is orthogonal to the grouping of *terms* into a *vocabulary*. *Term URIs* are grouped into *DCMI namespaces* in order to ease the assignment of URIs to *terms* and to streamline their use in particular encoding syntaxes. *Terms* are grouped into *vocabularies* in order to meet a functional need.

1. Introduction

All *terms* used in metadata descriptions that conform to the DCMI Abstract Model [DCAM] must be assigned a unique URI [RFC3986]. For convenience, the *term URIs* that are assigned and managed by the DCMI are grouped into collections known as *DCMI namespaces*. This document describes how *term URIs* are allocated by the DCMI and the policies associated with *DCMI namespaces*.

2. DCMI Namespace URIs

The *DCMI namespace URI* used for the collection of properties that make up the Dublin Core Metadata Element Set, Version 1.1 [DCMES] is:

<http://purl.org/dc/elements/1.1/>

The *DCMI namespace URI* used for the collection of all other DCMI properties and encoding schemes (i.e. other than those properties defined in the Dublin Core Metadata Element Set, Version 1.1 above) is:

<http://purl.org/dc/terms/>

The *DCMI namespace URI* used for the collection of *DCMI terms* 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 properties and encoding schemes (other than those properties defined in the Dublin Core Metadata Element Set, Version 1.1 above)
http://purl.org/dc/dcmitype/	DCMI terms in the DCMI Type Vocabulary

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

Some example *DCMI term URIs* follow:

<http://purl.org/dc/elements/1.1/title>

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

<http://purl.org/dc/terms/extent>

is the *DCMI term URI* for the Extent property, and

<http://purl.org/dc/dcmitype/Image>

is the *DCMI term 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/

3. 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 term URIs* and *DCMI namespaces*. The following classes of changes are identified along with examples and associated implications.

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 to *DCMI term 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 term 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 URI 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 URI* pattern defined above.

D. Addition of DCMI term declarations to existing DCMI namespaces

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

4. 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 *DCMI term URIs* and *DCMI namespace URIs* 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.

6. Justification

Two significant issues were raised during the development of this policy. Firstly, that *DCMI namespace URIs* should indicate the category of *DCMI terms* associated with that namespace. For example, it was proposed that different *DCMI namespaces* might be used to partition *DCMI properties* from *DCMI encoding schemes*, 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 *DCMI namespace URI* was unlikely to be helpful to the long-term stability of *DCMI namespace URIs* or *DCMI term URIs* within those *DCMI 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/> *DCMI namespace* could have been assigned new URIs within the <http://purl.org/dc/terms/> *DCMI namespace*, it was considered that the widespread existing usage of the former *DCMI namespace URI* mitigated against any change. Furthermore, the existing use of the purl.org domain for that *DCMI namespace URI* prompted its use for all *DCMI namespace URIs*.

References

[DCAM]

DCMI Abstract Model, DCMI Recommendation, March 2005

<http://dublincore.org/documents/abstract-model/>

[RFC3986]

IETF (Internet Engineering Task Force) RFC 3986: Uniform Resource Identifier (URI): Generic Syntax, eds T. Berners-Lee, R. Fielding, L. Masinter. January 2005

<http://www.ietf.org/rfc/rfc3986.txt>

[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, July 2000

<http://dublincore.org/documents/dcmi-type-vocabulary/>

[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>

Valid XHTML 1.0!	Valid CSS!
------------------	------------

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

Copyright © 1995-2005 DCMI All Rights Reserved. DCMI [liability](#), [trademark/service mark](#), [document use](#) and [software licensing](#) rules apply. Your interactions with this site are in accordance with our [privacy](#) statements. Please feel free to [contact us](#) for any questions, comments or media inquiries.

DCMI and the DCMI Web site are hosted by [OCLC Research](#).



[Home](#) > [Documents](#) > [Naming-policy](#) >

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>

Copyright © 1995-2005 DCMI All Rights Reserved. DCMI [liability](#), [trademark/service mark](#), [document use](#) and [software licensing](#) rules apply. Your interactions with this site are in accordance with our [privacy](#) statements. Please feel free to [contact us](#) for any questions, comments or media inquiries.

DCMI and the DCMI Web site are hosted by [OCLC Research](#).



[Home](#) > [Usage](#) > [Documents](#) > [Approval](#) >

Procedure for approval of DCMI Metadata Terms and Recommendations

Creator: [Makx Dekkers](#)

Date Issued: 2004-12-20

Identifier: <http://dublincore.org/usage/documents/2004/12/20/approval/>

Replaces: <http://dublincore.org/usage/documents/2003/08/14/approval/>

Is Replaced By: Not Applicable

Latest version: <http://dublincore.org/usage/documents/approval/>

Status of document This is a [DCMI Process Document](#)

Description of document: This document lists the step-by-step process for the approval of DCMI metadata terms and other recommendations.

Recent changes Shepherd of proposals related to metadata terms announces start of public comment period, not Usage Board chair; added link to detailed Usage Board process description.

Procedure for submission of proposals from DCMI Working Groups

Step	Event	Action	Result
1	Open issue or requirement is identified by the community and raised in a Working Group	Working Group (WG) Chair creates an Open Issue Item and adds it to WG Open issue list	Open Issue list change
2	Discussion on Working Group mailing list leads to proposal(s) for solution and identification of responsible authors/editorial team	WG Chair adds the deliverable to the WG Task List, including authors/editorial team responsible for the document and a target delivery date	WG Task list change
3	Authors/editorial team finalize draft document	Authors/editorial team (in consultation with WG Chair) post message to WG and link document from WG pages	DCMI Working Draft
4	WG Discussion	WG Chair manages the discussion and iterative review, and coordinates revision of proposal with the authors/editorial team	DCMI Working Draft (revisions)
5	Consensus reached in WG	WG Chair summarizes consensus and posts last call to WG	DCMI Working Draft (final)
6	Resolution of all comments	WG Chair submits to Managing Director	DCMI WG Proposal
7	DCMI Managing Director receives DCMI WG Proposal	DCMI Managing Director assigns WG Proposal to review team (Usage Board if related to Metadata Term Semantics or (subset of) Advisory Board plus external reviewers otherwise).	Review team established

Continuation of procedure for review and recommendations for technical proposals

Step	Event	Action	Result
8	Review starts	DCMI Managing Director assigns a shepherd to the proposal	DCMI WG Proposal Review in process
9	Review team discussion	Proposal shepherd manages the discussion in the review team and coordinates Review comments	Review Team discussion results
10	Review Results submitted to DCMI Managing Director	DCMI Managing Director and Review Team Chair, in consultation with proposal editor, evaluate Review Team discussion results and decide to either accept (possibly with changes) or reject the proposal	If accepted, proposal becomes a <i>DCMI Proposed Recommendation</i> If rejected, proposal is referred back to the WG for further discussion
11	Final text for <i>DCMI Proposed Recommendation</i> available	DCMI Managing Director posts DCMI Proposed Recommendation to DC-General and DCMI Web site for Public Comment	Public Comment period commences
12	Public Comment period (minimum four weeks)	Proposal shepherd manages Public Comment period	Public Comments from community
13	Public Comment period finishes	Review team evaluates Proposal and Public Comment results and recommend approval (possibly with changes) or rejection with cause	Recommendation for approval or rejection to DCMI Managing Director
14	DCMI Directorate receives recommendation	DCMI Managing Director, in consultation with WG Chair, review team and others, decides on outcome of process and makes public announcement	If accepted, proposal becomes <i>DCMI Recommendation</i> If rejected, special action
15	Final text for <i>DCMI Recommendation</i> available	DCMI Managing Director posts <i>DCMI Recommendation</i> to DC-General and DCMI Web site	Process completed

Continuation of procedure for review and acceptance of proposals related to metadata terms

For a more detailed description, see the [DCMI Usage Board Administrative Processes document](#).

Step	Event	Action	Result
8	Review starts	Usage Board chair assigns a shepherd to the proposal; proposal is posted on DCMI Web site for Public Comment; shepherd announces to DC-General	Public Comment period commences
9	Public Comment period (minimum four weeks)	Proposal shepherd manages Public Comment period	Public Comments from community
10	Public Comment period finishes	Proposal shepherd prepares for Usage Board discussion	Summary of Public Comments and proposed resolution for Usage Board
11	Usage Board meeting or conference call	Usage Board discusses proposal, taking into account Public Comment results, according to established guidelines and criteria and recommend approval or rejection with cause	Usage Board recommends approval or rejection to DCMI Managing Director

12	DCMI Managing Director receives recommendation	DCMI Managing Director, in consultation with Usage Board Chair and others, decides on outcome of process and makes public announcement	If accepted, proposed term becomes part of <i>DCMI Metadata Terms</i> If rejected, proposal is referred back to the WG for further discussion
13	Proposal accepted	Usage Board Chair prepares new version of authoritative documentation; Web team loads new term into http://dublincore.org/documents/dcmi-terms/	Process completed



Metadata associated with this resource: <http://dublincore.org/usage/documents/approval/index.shtml.rdf>

Copyright © 1995-2005 DCMI All Rights Reserved. DCMI [liability](#), [trademark/service mark](#), [document use](#) and [software licensing](#) rules apply. Your interactions with this site are in accordance with our [privacy](#) statements. Please feel free to [contact us](#) for any questions, comments or media inquiries.

DCMI and the DCMI Web site are hosted by [OCLC Research](#).



Dublin Core Metadata Initiative®

- ABOUT THE INITIATIVE
- DCMI NEWS
- DOCUMENTS
- TOOLS AND SOFTWARE
- GROUPS
- PROJECTS
- RESOURCES
- AskDCMI

[Home](#) > [Documents](#) > [Abstract-model](#) >

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](#)
 - [2. DCMI abstract model](#)
 - [3. Descriptions, description sets and records](#)
 - [4. Values](#)
 - [5. Dumb-down](#)
 - [6. Encoding guidelines](#)
 - [7. Terminology](#)
- [References](#)
[Acknowledgements](#)
[Appendix A - A note about structured values](#)
[Appendix B - The abstract model and RDF](#)
[Appendix C - The abstract model and XML](#)
[Appendix D - The abstract model and XHTML](#)

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.

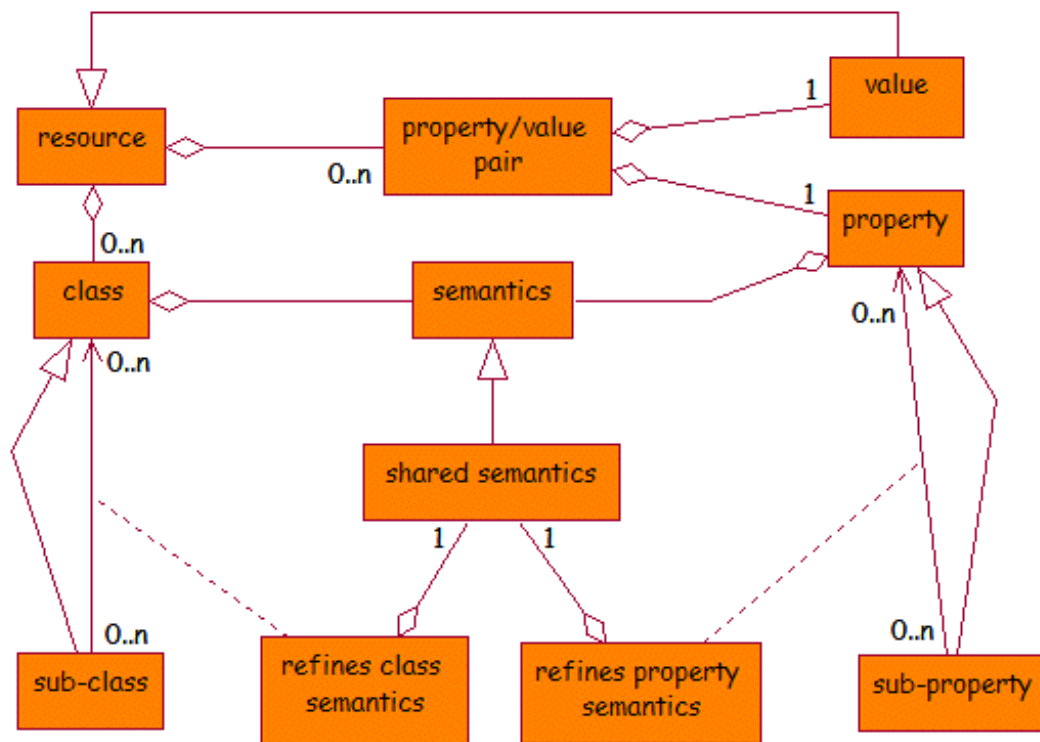


Figure 1 - the DCMI resource model



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

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:

<i>class</i>	A <i>class</i> is a group containing members that have attributes, behaviours, relationships or semantics in common; a kind of category.
<i>class URI</i>	A <i>class URI</i> is a URI reference that identifies a <i>class</i> .
<i>description</i>	A <i>description</i> is made up of one or more <i>statements</i> about one, and only one, <i>resource</i> .
<i>description set</i>	A <i>description set</i> is a set of one or more <i>descriptions</i> about one or more <i>resources</i> .
<i>element</i>	Within DCMI, <i>element</i> is typically used as a synonym for <i>property</i> . However, it should be noted that the word <i>element</i> is also commonly used to refer to a structural markup component within an XML document.
<i>element refinement</i>	An <i>element refinement</i> is a <i>property</i> of a <i>resource</i> that shares the meaning of a particular DCMI <i>property</i> but with narrower semantics. Since <i>element refinements</i> are <i>properties</i> , they can be used in metadata <i>descriptions</i> independently of the <i>properties</i> they refine. In DCMI practice, an <i>element refinement</i> refines just one parent DCMI <i>property</i> .
<i>encoding scheme</i>	<i>Encoding scheme</i> is the generic name for <i>vocabulary encoding scheme</i> and <i>syntax encoding scheme</i> .
<i>encoding scheme URI</i>	The generic name for a <i>vocabulary encoding scheme URI</i> or a <i>syntax encoding scheme URI</i> .
<i>marked-up text</i>	A string that contains HTML, XML or other markup (for example TeX) and that is associated with the <i>value</i> of a <i>property</i> .
<i>property</i>	A <i>property</i> is a specific aspect, characteristic, attribute, or relation used to describe <i>resources</i> .
<i>property URI</i>	A <i>property URI</i> is a URI reference that identifies a single <i>property</i> .
<i>property/value pair</i>	A <i>property/value pair</i> is the combination of a <i>property</i> and a <i>value</i> , used to describe a <i>resource</i> .
<i>qualifier</i>	<i>Qualifier</i> was the generic name used for the <i>terms</i> that are now usually referred to specifically as <i>element refinements</i> or <i>encoding schemes</i> .
<i>record</i>	A <i>record</i> is a <i>description set</i> that is instantiated according to one of the DCMI encoding guidelines (XHTML meta tags, XML, RDF/XML, etc.)
<i>related description</i>	

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:

- o 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).
- o Some *marked-up text*.
- o 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.

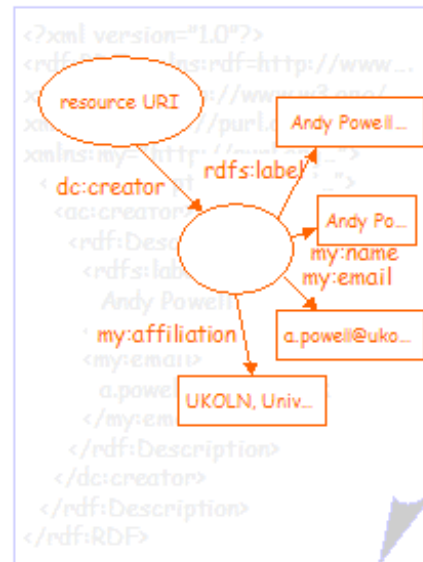


Figure 3

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.

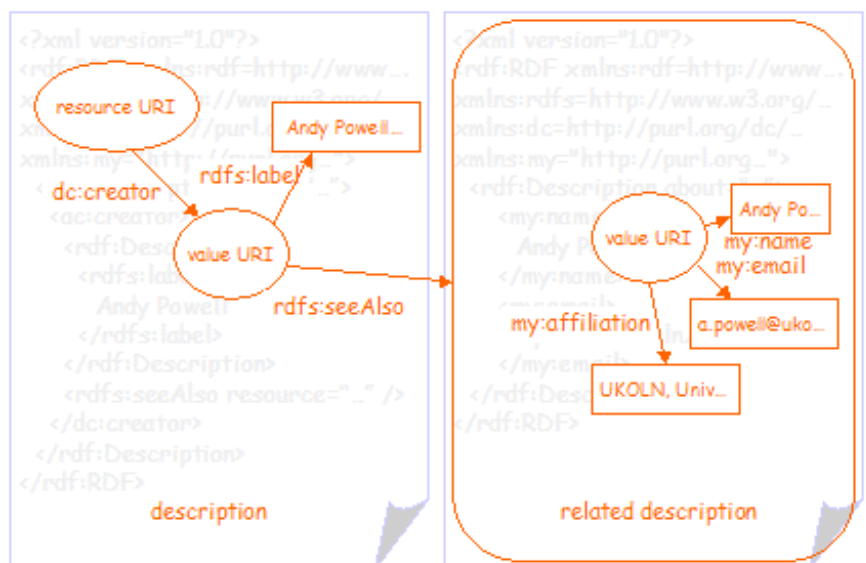


Figure 4

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.

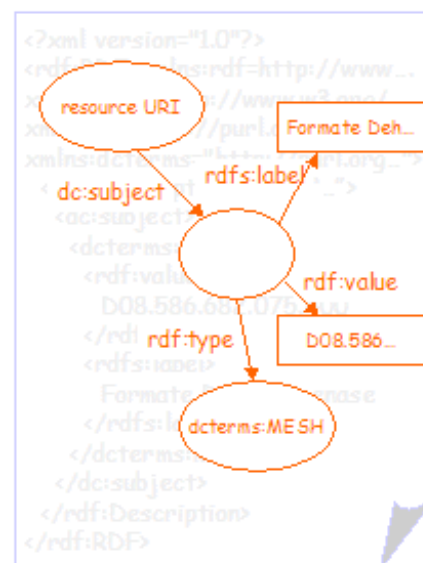


Figure 5

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.

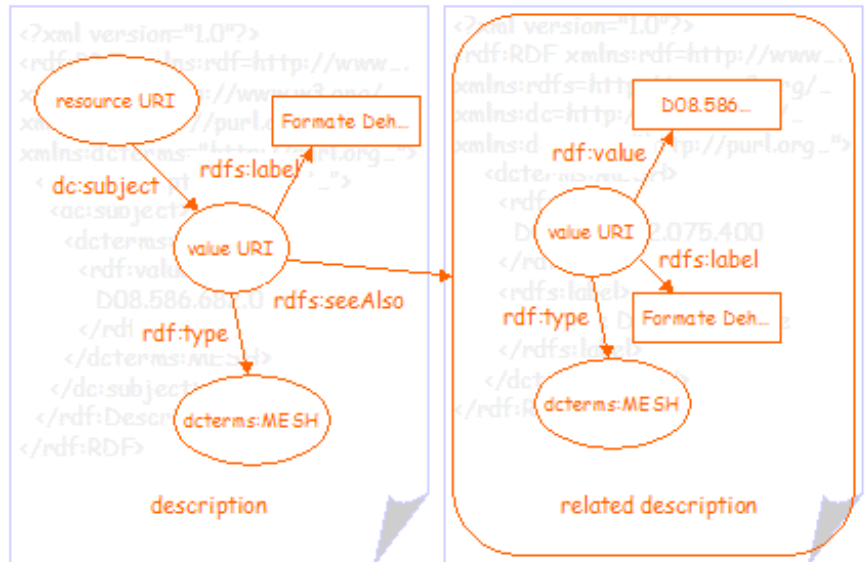


Figure 6

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.

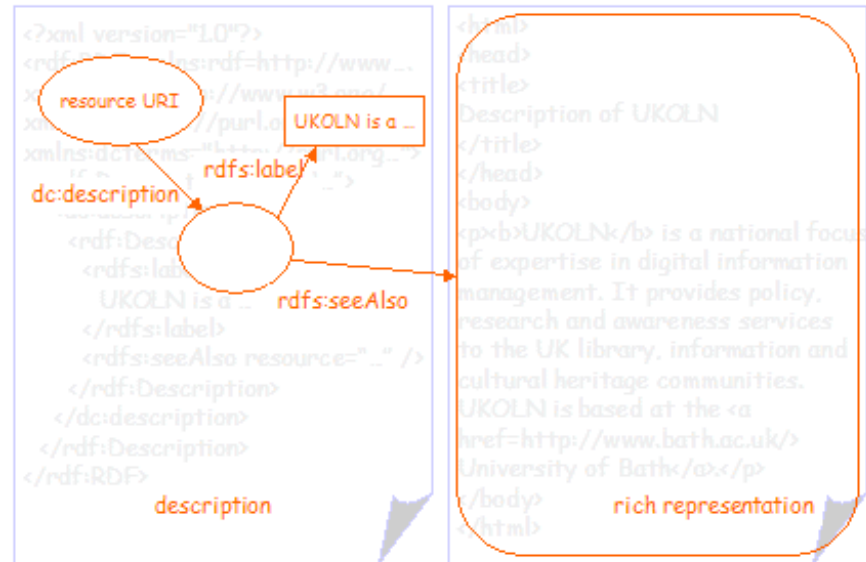


Figure 7

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.

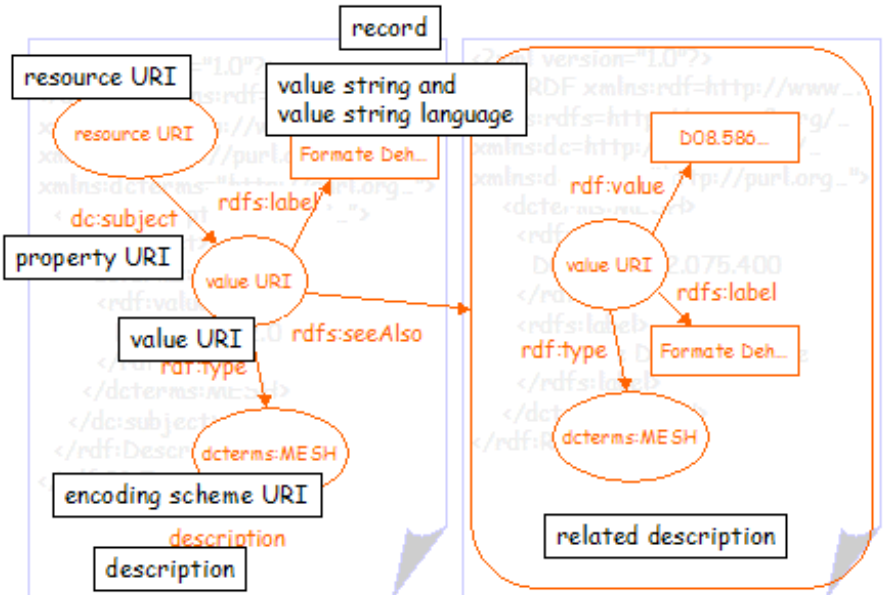


Figure 8

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

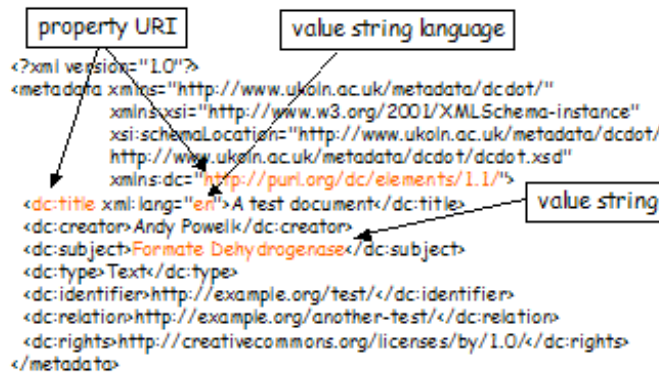


Figure 9

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

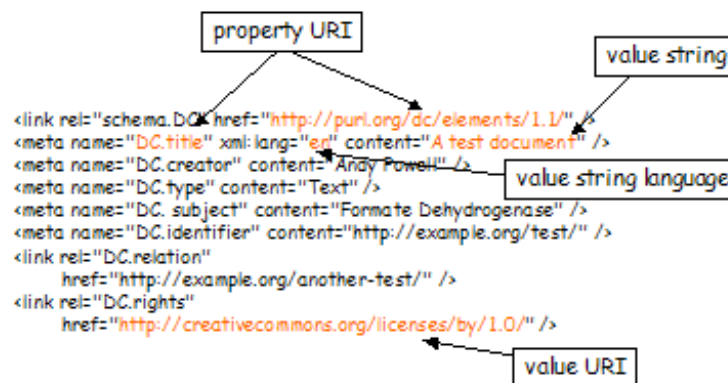


Figure 11

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

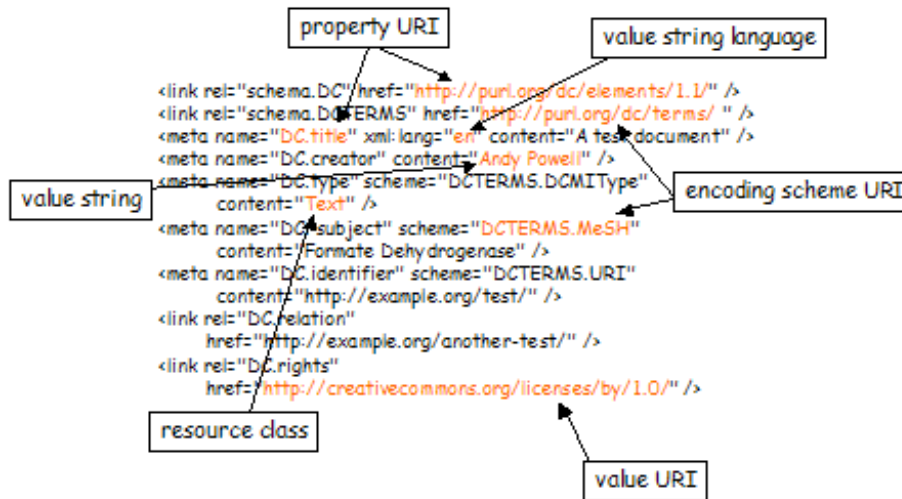


Figure 12

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.



Metadata associated with this resource: <http://dublincore.org/documents/abstract-model/index.shtml.rdf>

Copyright © 1995-2005 DCMI. All Rights Reserved. DCMI [liability](#), [trademark/service mark](#), [document use](#) and [software licensing](#) rules apply. Your interactions with this site are in accordance with our [privacy](#) statements. Please feel free to [contact us](#) for any questions, comments or media inquiries.

DCMI and the DCMI Web site are hosted by [OCLC Research](#).

Dublin Core Metadata Initiative logo	About the Initiative	Documents	Working Groups	Resources
	Dublin Core Metadata Initiative	Tools and Software	Meetings	Projects
Dublin Core Metadata Initiative				

[Home](#) > [Documents](#) >

Title:	Guidelines for assigning identifiers to metadata terms
Creator:	Andy Powell UKOLN, University of Bath, UK
Date Issued:	2004-08-01
Identifier:	http://www.ukoln.ac.uk/metadata/dcmi/term-identifier-guidelines/
Replaces:	
Is Replaced By:	Not applicable
Latest Version:	http://www.ukoln.ac.uk/metadata/dcmi/term-identifier-guidelines/
Status of Document:	This is a DRAFT DCMI Recommended Resource.
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 DCMI Abstract Model [\[DCMI-AM\]](#) 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 [\[RFC3986\]](#) that identifies the term. An XML namespace [\[XML-NAMES\]](#) is a collection of names, identified by a URI, 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 and the term name to provide a mechanism for encoding the term URI. The use of XML namespaces and URI 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 URIs 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 DCMI 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. The use of fragment identifiers in the URI used to identify metadata terms is optional and is left to the discretion of the implementor.

For the purposes of encoding, the term URI may be partitioned into an XML namespace URI and the term name. Note that, for convenience, it is commonly the case that XML namespace URIs end with either a '#' (hash) or '/' (slash) character.

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

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

Any valid URI [\[RFC3986\]](#) 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 URIs should be assigned with the intention of them being unique and persistent. This means that the URI must not be used to identify anything else and that it should be expected to last as long as the Internet.

3. Strategies for assigning URIs

Four simple strategies for assigning URIs 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 within its existing URL space (<http://example.org/terms/>) and therefore assigns the term the following URI: <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 within their existing URL space (<http://myproject.org/metadata/vocabs/color#>). For the vocabulary term `Red`, the term URI 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 URIs, 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 against 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 <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 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. 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 URIs 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 URIs to information assets that have identifiers in public namespaces"* but that do not have an appropriate existing URI scheme [\[INFO-URI-SPEC\]](#) [\[INFO-REGISTRY\]](#). 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 and "004.678" can be considered to be a Dewey term name. Thus the URI 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). Indeed, "info" URIs are designed to be non-dereferencable - i.e. it is not possible to dereference an "info" URI in order to retrieve a representation of the identified resource. Unfortunately, this has serious consequences on their utility for identifying metadata terms. Since it is not possible to easily obtain a representation of the identified term (typically some metadata about the term), it is not possible to obtain any information about the relationships between the identified term and other terms. This means that the "info" URI is of limited use in the context of the Semantic Web, since it is not possible for software applications to reason automatically based on knowledge about the relationships between multiple metadata terms.

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 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 and has been assigned the URI `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 URIs based on this domain.

4. Conclusions

All terms used in metadata application profiles must be assigned a URI before they can be used in the encoding syntaxes recommended by DCMI. It is recommended that implementors assign URIs to terms following the guidelines provided here. Of the four strategies for assigning URIs to terms listed in this document, the use of PURLs is recommended for the identification of all metadata terms.

References

[DCMI-AM]

DCMI Abstract Model

<http://dublincore.org/documents/abstract-model/>

[XML-NAMES]

Namespaces in XML, W3C Recommendation, 14 January 1999

<http://www.w3.org/TR/REC-xml-names>

[RFC3986]

IETF (Internet Engineering Task Force) RFC 3986: Uniform Resource Identifiers (URI): Generic Syntax, T. Berners-Lee, R. Fielding, L. Masinter. January 2005.

<http://www.ietf.org/rfc/rfc3986.txt>

[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/>

Valid XHTML 1.0!	Valid CSS!
------------------	------------

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

[Copyright](#) © 1995-2002 [DCMI](#) All Rights Reserved. DCMI [liability](#), [trademark/service mark](#), [document use](#) and [software licensing](#) rules apply. Your interactions with this site are in accordance with our [privacy](#) statements. Please feel free to [contact us](#) for any questions, comments or media inquiries.

Title:	XML, RDF, and DCAPs
Creator:	Pete Johnston
Date Issued:	2005-02-17
Identifier:	http://www.ukoln.ac.uk/metadata/dcml/dc-elem-prop/2005-02-17/
Replaces:	Not applicable
Is Replaced By:	Not applicable
Latest Version:	http://www.ukoln.ac.uk/metadata/dcml/dc-elem-prop/
Description of Document:	This document describes the differences between XML and RDF, and between DC elements, XML elements and RDF properties. It seeks to clarify the requirements that must be met before a "term" can be referenced in a Dublin Core Application Profile (DCAP).

Contents

1. [Introduction](#)
2. [XML, XML Elements, XML Namespaces and XML Languages](#)
3. [RDF, URI references and RDF/XML](#)
4. [XML and RDF](#)
5. [Dublin Core and Dublin Core Application Profiles](#)
6. [Conclusions and Recommendations](#)
7. [Notes](#)
8. [References](#)

1. Introduction

For some time DCMI has advocated the approach that the terms of the Dublin Core metadata vocabularies can be deployed in combination with similar terms defined by other sources. This has led to the development of the concept of the "Dublin Core Application Profile" (DCAP) [[DCAPUB](#)], as a specification which:

- lists the terms that are used within a class of DC metadata descriptions
- (optionally) describes constraints on how those terms are used within those DC metadata descriptions

A DCAP may be specific to a single application, or it may reflect the usage of an implementer community.

Although the DCMI Usage Board has procedures in place to "review" DCAPs that are submitted to it for evaluation, DCMI does not currently have a formal specification for what constitutes a DCAP. The CEN CWA that described how to present a human-readable representation of a DCAP [[CWA14855](#)] described a fairly "permissive" notion of a DCAP. Perhaps as a consequence, implementer interpretations of the concept have tended to vary somewhat; in particular, there appears to be some divergence amongst DCAP designers regarding the nature of the "terms" that are referenced or "used" within a DCAP.

The DCMI Abstract Model [[DCMIAM](#)] describes a conceptual framework for Dublin Core metadata descriptions: it describes the logical components which make up DC metadata descriptions and the relationships between them. Although the notion of the DCAP is not explicitly addressed within the Abstract Model, if a DCAP is to specify how a particular set of DC metadata descriptions are constructed, then it follows that the types of "term" referenced within a DCAP must correspond to the types of component described within the Abstract Model.

This document examines some of the specifications used for the representation of data, and particularly the data models used within those specifications. It seeks to clarify some of the terminology and concepts used within those specifications, and in particular to highlight significant differences between concepts that may at first appear to be similar.

It concludes by returning to the question of the DCAP and makes some suggestions on what is required to provide "terms" that *are* usable in DC metadata descriptions, and so are appropriate for reference from a DCAP.

Note: This document provides a good deal of technical background information. It is intended principally for the DCMI Usage Board, rather than as a document for general circulation. Once agreement is reached about the nature of the problem and possible approaches to solving it, then a more concise, user-oriented summary of recommendations for good practice might be produced.

2. XML, XML Elements, XML Namespaces and XML Languages

2.1 XML

The XML 1.0 specification [[XML](#)] defines a means of describing structured data in a text-based format. XML uses **tags** embedded in the content of a document to delimit and label parts of the document, and those parts are known as **XML elements**. Tags themselves begin and end with special characters (<...>) so that they can be distinguished from the **element content**, and XML element **end tags** can be distinguished from **start tags** by a special character combination (</...).

The start and end tags include an **XML element type name** and may also contain **XML attributes** (see below). XML elements may contain character data (only), other XML elements, a combination of character data and XML elements - or nothing, i.e. XML elements can be empty. (See [Note 1.](#))

An **XML attribute** is a pair made up of an **attribute name** and an **attribute values**. Multiple XML attributes can occur within the start tag of an element, but each start tag can contain only one XML attribute with a given attribute name. XML attribute values can contain only character data.

```
<?xml version="1.0"?>
<metadata>
  <title lang="en">DCMI Home Page</title>
</metadata>
```

Example 1

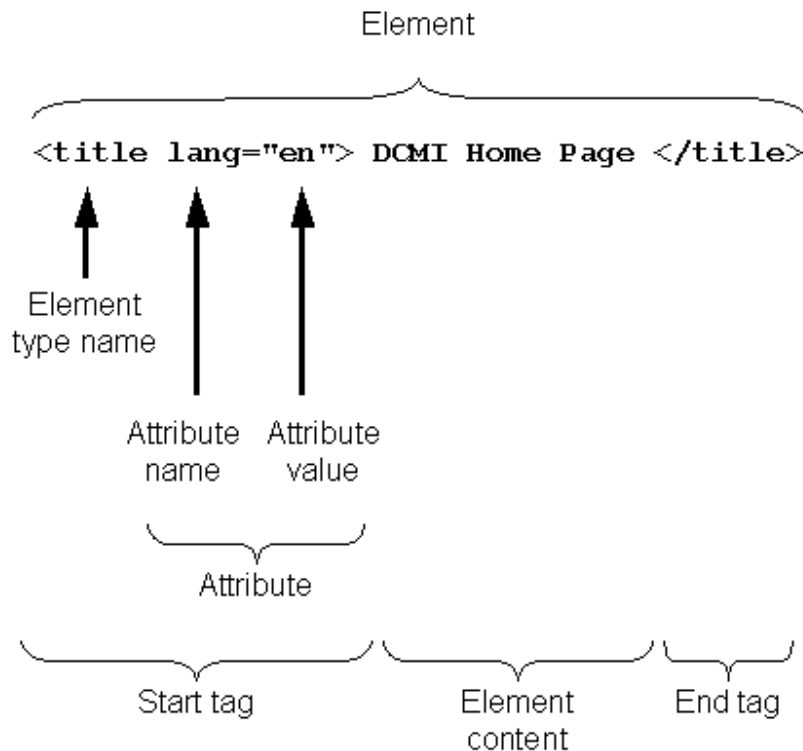


Figure 1: XML Elements and XML Attributes

This document uses the term **component** to refer to XML elements and XML attributes.

XML does not provide a fixed set of element type names and attribute names. Rather users of XML define their own sets of element type names and attribute names for use in tags in XML documents. For this reason, XML is sometimes referred to as a "**meta-language**", a set of rules for defining XML languages.

2.2 XML DTDs and XML Schemas

XML Document Type Definitions (DTDs) [XML] and XML Schemas [XMLS] provide means of describing/defining constraints on the **structure** of a class of XML documents, the structural relationships that can exist between components: for a named XML element type, the names of the child XML elements it can contain and the XML attributes can be associated with it, and so on. i.e. XML Schemas and XML DTDs describe **content models** for named XML element types and attributes. XML Schema also introduces a datotyping mechanism which is not discussed further in this document.

An XML document which conforms to the rules of the XML specification and to the structural constraints described by an XML DTD or XML Schema is described as **valid**.

An XML document is described as **well-formed** if it meets certain syntactic constraints: simplifying slightly, well-formedness requires that the document contains only one outermost XML element (the root element), that each XML element has a start and end tag, and that tags are not overlapping. An XML document can be well-formed without being associated with an XML DTD or XML Schema.

2.3 XML Namespaces and XML Qualified Names

As noted above, users of XML define sets of element type names and attribute names for use in tags in XML documents. Further, it can be useful to (re)use independently defined sets of names in combination within the same XML document. However, this raises the prospect of collisions between names which have been defined in multiple name sets.

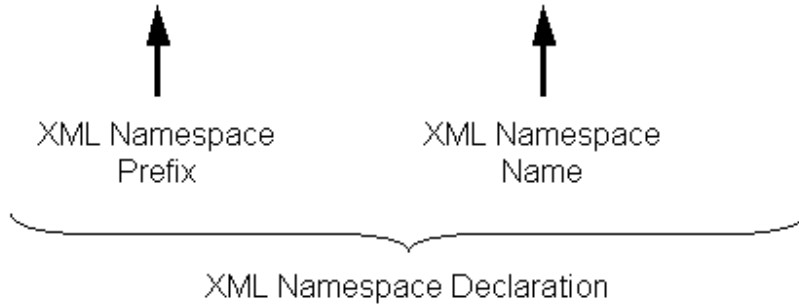
The *Namespaces in XML* specification [XMLNS] seeks to address the problem of name collisions by providing a mechanism for giving **expanded names** to XML elements and XML attributes. An expanded name is a pair made up of two parts: an **XML Namespace Name** (which is a URI reference) and a **local name**. N.B. An expanded name is *not* itself a URI reference.

Namespaces in XML also introduces the **XML Qualified Name (QName)** as a syntactic construct for deploying expanded names in XML documents. A QName consists of a prefix and a local part. Namespaces are applied to XML elements and XML attributes through the mechanism of a **namespace declaration** which applies to all XML element and XML attribute names within its scope which have a prefix that matches that specified in the declaration. The namespace declaration is said to "bind" a prefix to an XML Namespace Name.

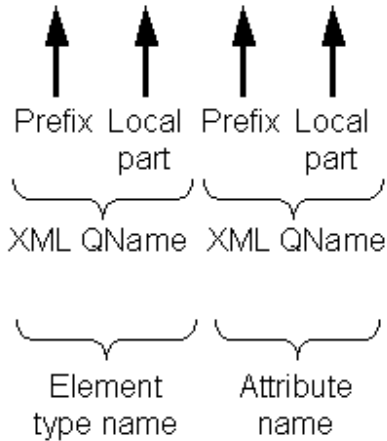
```
<?xml version="1.0"?>
<metadata xmlns:dc="http://purl.org/dc/elements/1.1/">
  <dc:title xml:lang="en">DCMI Home Page</dc:title>
</metadata>
```

Example 2

```
<metadata xmlns:dc="http://purl.org/dc/elements/1.1/">
```



```
<dc:title xml:lang="en">DCMI Home Page</dc:title>
```



```
</metadata>
```

Expanded Names:	
Namespace Name:	http://purl.org/dc/elements/1.1/
Local Name:	title
Namespace Name:	http://www.w3.org/XML/1998/namespace
Local Name:	lang

Figure 2: XML Namespaces

(The prefix xml is reserved and does not require an XML Namespace declaration; it is bound to the namespace name <http://www.w3.org/XML/1998/namespace>).

An **XML Namespace** is a collection of names of XML element types and attributes. N.B. It is only a collection of **names**, not a collection of XML elements and attributes. Further, within a single XML document, the same expanded name may be used as both an XML element type name and an XML attribute name (e.g. in an RDF/XML document the expanded name ("http://purl.org/dc/elements/1.1/", "title") may be used, encoded as an XML QName `dc:title`, as both an XML element type name and an XML attribute name.

It is important to note that the XML Namespaces specification provides *only* a means of disambiguating the *names* of components in an XML document: the XML Namespaces specification does *not* provide a basis for "merging" together two XML documents. This is discussed further in the next two sections.

2.4 The XML Infoset: the XML "Abstract Model"

A well-formed XML document can be represented as a **tree structure**, and the XML Information Set [[XMLINFO](#)] is an abstract model that describes the set of **information items** of different types which are available from any well-formed XML document. Conversely, any well-formed XML document can be viewed simply as a representation of an XML Information Set.

For example, this XML document

```
<?xml version="1.0"?>
<metadata xmlns:dc="http://purl.org/dc/elements/1.1/">
  <dc:title xml:lang="en">DCMI Home Page</dc:title>
  <dc:description xml:lang="en">DCMI is an open forum engaged in the
  development of interoperable online metadata standards.</dc:description>
  <dc:publisher>DCMI</dc:publisher>
  <dc:subject>metadata</dc:subject>
  <dc:subject>resource discovery</dc:subject>
</metadata>
```

Example 3

would be represented as the following tree of XML InfoSet items. (See [Note 2](#))

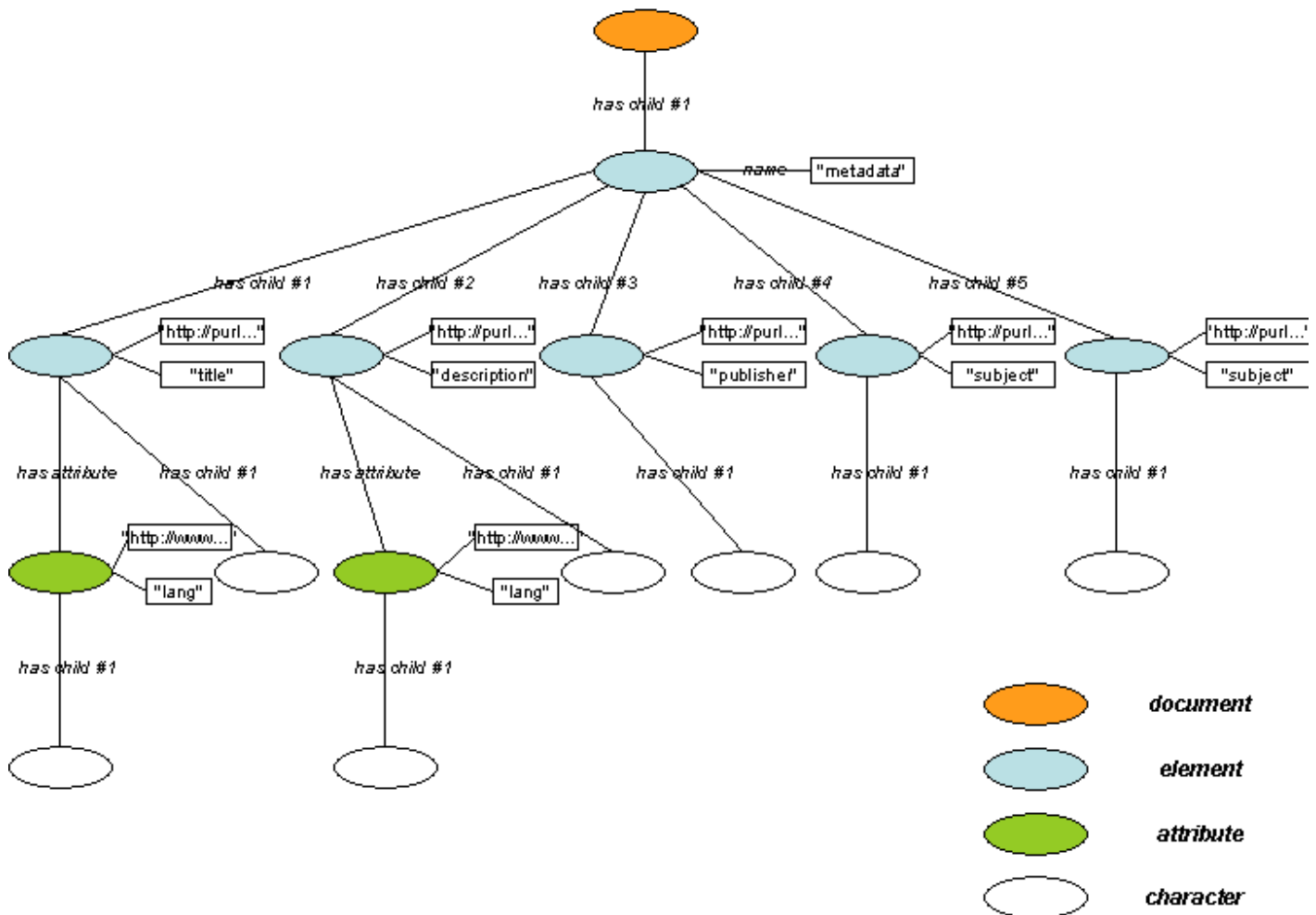


Figure 3: The XML Infoset

Note that:

- the types of relationships between items are defined by the XML Infoset and are (at least for the subset of items considered here) parent-has-child/child-has-parent and element-has-attribute/attribute-is-owned-by-element relationships.
- the Infoset has one root/document item, and each other item has exactly one parent (for an element) or owner (for an attribute)
- the order of the child information items is represented
- information items are not uniquely identified and several information items are associated with the same expanded name (i.e. the two `<dc:subject>` elements). Information items can be uniquely addressed, but only by some reference to their context in the tree structure, their relationship to other items

Although the XML Infoset is not a specification for an application programme interface, and XML APIs typically do not present all the information described by the XML Infoset specification, most XML parsers present this type of "view" of an XML document. Similarly specifications like XPath (for addressing parts of an XML document) [[XPath](#)] and XQuery (for querying XML documents) [[XQuery](#)] are based on a tree view of the document.

Further, although the XML Infoset specification was created after the XML specification, it is possible to take the view that any XML document is simply a serialisation of an XML Infoset, and a number of XML-based specifications are defined with reference to the XML Infoset rather than to the XML specification. It may be helpful to try to think in terms of the XML Infoset, or at least of a tree structure, rather than the text syntax, when making comparisons between XML and RDF (see below).

XML itself says nothing about the intended meaning of element type names and attribute names. Furthermore, information is also conveyed by the structural relationships between components within XML documents, such as the parent-child relationships between nested XML elements or element-attribute relationships. XML does not prescribe any fixed meaning for those structural relationships, and in different XML applications, the same structural relationship may carry quite different meanings.

Even within the same application, the same structural relationship may carry a different significance at different contexts within the tree structure.

Those meanings are typically described in human-readable documents which specify how a particular set of named XML elements types and attributes are to be interpreted. (See below on XML languages).

So for example, suppose the designer of an XML application wants to represent the information that the document with the title "Progress Report" was authored by an entity named "John Smith". They might choose any of the following XML structures:

```
<?xml version="1.0"?>
<my:metadata xmlns:my="http://example.org/my/">
  <my:title>Progress Report</my:title>
  <my:author>John Smith</my:author>
</my:metadata>
```

Example 4

```
<?xml version="1.0"?>
<your:metadata xmlns="http://example.org/your/"
  your:title="Progress Report"
  your:author="John Smith"/>
```

Example 5

```
<?xml version="1.0"?>
<his:metadata xmlns:his="http://example.org/his/">
  <his:general>
    <his:title>Progress Report</his:title>
  </his:general>
  <his:lifecycle>
    <his:author>John Smith</his:author>
  </his:lifecycle>
</his:metadata>
```

Example 6

```
<?xml version="1.0"?>
<her:metadata xmlns:her="http://example.org/her/">
  <her:x>
    <her:t>Progress Report</her:t>
  </her:x>
  <her:y her:a="John Smith"/>
</her:metadata>
```

Example 7

All of these are good uses of XML, but they result in very different XML Infosets. It is impossible to interpret what meaning is being conveyed in any of these documents unless the author of the document or the designer of the XML application provides a description which explains what the names of the components and the structural relationships between those components are intended to convey. A human reader (or at least an English-speaking one!) may be tempted to guess at the interpretation based on the names of the components, but as the last example illustrates, XML imposes no requirement that names are drawn from human languages.

Similarly a software application querying these documents would have to be programmed to navigate the four different tree structures. The question "What is the name of the author of the work titled 'Progress Report'?" must be translated into a different query on the tree structure in each case.

2.5 Names in XML Vocabularies and XML Languages

Effective information exchange using XML depends on the sender and receiver of the XML document having a common understanding of the meaning conveyed by the names used in the XML document and by the structural relationships between named components in the XML document. That is, information exchange depends on the shared use of **XML languages** (or **formats**) and on the sender and receiver having a common understanding of the rules of the XML language. All XML documents are instances of XML languages, and the interpretation of an XML document is determined by the specification of an XML language.

Such an **XML language** or format has three parts:

1. An **XML vocabulary**: a set of names, drawn from one or more XML Namespaces (or from none), which are used as XML element type names and XML attribute names.
2. A set of **structural constraints** which specifies how the names are to be used as names of components, and describes the permitted content models for those components. The constraints may be expressed by an XML DTD or XML Schema or by some other formalism, or may simply take the form of a narrative description.
3. A **description** of how the named components, and the structural relationships between those components, are to be interpreted to convey information. (see [Note 3](#))

[image to follow]

Figure 4: XML Vocabularies and XML Languages

It is worth exploring some facets of the complex relationships between names, vocabularies, and languages.

2.5.1 XML languages do not require the use of XML Namespaces

Many XML languages do not make use of XML Namespaces in their vocabularies. Examples include Docbook and Encoded Archival Description (EAD). These two XML languages may include components with the same names, but those components have different content models and the meaning conveyed by those components is different (and is described by the human-readable language specifications).

2.5.2 XML Vocabularies and XML Namespaces

There is no simple correspondence between between the set of names used in an XML vocabulary and an XML Namespace. A vocabulary may draw on names that are associated with multiple XML Namespace Names. And the vocabularies of different XML languages may utilise different sets of names associated with the same XML Namespace. e.g. the XHTML 1.0 specification defines three different XML languages: XHTML Transitional, XHTML Strict and XHTML Frameset. Each uses a *different* XML vocabulary but in each case the set of names is associated with the *same* XML Namespace Name `http://www.w3.org/1999/xhtml`.

2.5.3 One name, one component, different constraints

A *single* name may be used as the name of an XML component (XML element, XML attribute) in *multiple* XML languages, and the named component may be associated with a *different* set of structural constraints in each XML language, e.g. the XML vocabulary of the XHTML Transitional language is a superset of the XML vocabulary of the XHTML Strict languages. However in each of those languages the named components are associated with a different set of structural constraints, different content models.

2.5.4 One name, multiple components

Within a single XML language, a *single* name (whether it is qualified by an XML Namespace Name or not) may be associated with *different* types of XML component. The information conveyed by those different components may be different, even if their names are the same.

For example, XHTML uses the name `link` as the name of both an XML element and an XML attribute, but the information conveyed by those two components is quite different.

[This is actually not a good example as the name of the attribute is not namespace qualified so the name of the element is different from the name of the attribute! But the principle holds! I'll try to find a better example.]

2.5.5 One name, one component, different contexts

Within a single XML language, the way individual components are interpreted is conditioned by their structural relationships with other components (containment relations, element/attribute relations etc). So the same name may occur as the name of a component in different contexts in the tree-structure, and it may convey different meaning in those two contexts. For example, in the XML format used to represent instances of the IEEE Learning Object Metadata standard, an XML element with the expanded name "`http://`" (typically represented by the QName `lom:language`) may occur in three different contexts in the XML tree structure:

```
<?xml version="1.0"?>
<lom:lom xmlns:lom="http://ltsc.ieee.org/xsd/LOM">
  <lom:general>
    <lom:language>en</lom:language>
  </lom:general>
  <lom:metametadata>
    <lom:language>en</lom:language>
  </lom:metametadata>
  <lom:educational>
    <lom:language>en</lom:language>
  </lom:educational>
</lom:LOM>
```

Example 8

The same XML element conveys three different pieces of information depending on its context in the tree structure:

- as a child element of the `lom:general` XML element, it is used to represent the language used within the learning object
- as a child element of the `lom:metametadata` XML element, it is used to represent the language of the metadata instance
- as a child element of the `lom:education` XML element, it is used to represent the language of a typical user of the learning object

2.5.6 Ordering of components

A variant of the previous case of context conditioning interpretation is that the ordering of components may be significant in an XML language. In the LOM XML binding, ordering is considered significant in several parts of the tree-structure e.g. a sequence of source/value XML element pairs is used to represent a list of learning resource types, and according to the LOM standard, "The most dominant kind shall be first".

2.5.7 One name, different components, different languages

Consider the following three documents:

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:dc="http://purl.org/dc/elements/1.1/">
  <rdf:Description dc:title="DCMI Home Page" />
</rdf:RDF>
```

Example 9: RDF/XML

```
<?xml version="1.0"?>
<description xmlns:dc="http://purl.org/dc/elements/1.1/">
  <dc:title>DCMI Home Page</dc:title>
</description>
```

Example 10: DC-XML

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:dc="http://purl.org/dc/elements/1.1/">
  <xsl:template match="/">
    <dc:title>
      <xsl:text>DCMI Home Page</xsl:text>
    </dc:title>
  </xsl:template>
</xsl:stylesheet>
```

Example 11: XSLT

The XML QName `dc:title` occurs as the name of a component in all three XML documents, and in each case it corresponds to the same expanded name ("<http://purl.org/dc/elements/1.1/>", "title").

In [Example 9](#), it is the name of an XML attribute, and when the XML document is interpreted following the rules of the RDF/XML specification [[RDFXML](#)], the XML Infoset is interpreted as representing a single RDF triple.

In [Example 10](#), it is the name of an XML element, with text only content, and if the XML document is interpreted following the rules of the DC-XML language described in *Guidelines for Expressing DC in XML* [[DCXML](#)] [assuming they were better written!], the document is interpreted as representing a Dublin Core metadata description, consisting of a single statement about an unidentified resource.

In [Example 11](#), it is again the name of an XML element, but this time with a single child XML element. The document is interpreted following the rules of the XSLT XML language [[XSLT](#)], and the expanded name ("<http://purl.org/dc/elements/1.1/>", "title") is interpreted not as part of a DC metadata description, but simply as the name of an element node to be added to the XML result tree generated by the XSLT transformation.

2.5.8 Summary

The aim of providing these detailed examples is to illustrate that the same XML expanded name (XML Namespace Name, Local Name pair) may be used in **multiple** XML vocabularies and in **multiple** XML languages. While the XML Namespaces specification provides for the avoidance of name collisions, it does not address the question of what it means to mix named components from different XML languages. XML components (XML elements and XML attributes) are not, in the general case, "stand-alone" and can not be interpreted independently of their context in an XML document. Meaning in XML documents is derived from combinations of named components. There is some expectation that the name has a consistent meaning across XML languages, but the meaning of a named component is always **scoped by the XML language in which it occurs**, and even within a single XML language, the meaning of a named component may be dependent on **the context of that component within the tree structure**.

2.6 Modularity and Extensibility in XML Languages

The ability to use components of an XML language independently of other components of the language and to (re)use components that are specified within one XML language in the context of another XML language should not be taken for granted. If an XML language is to be extensible, that extensibility must be built into the design of the language.

Some XML languages are defined as essentially standalone and are intended to be used more or less by themselves (e.g. TEI, XHTML). But some XML languages are created to be used in association with other languages.

Some are container languages where the expectation is that they will act as a wrapper for components (sometimes referred to as a "payload") which are themselves constructed according to the rules of another language, where this second language may not even be known at the time the container language is designed. Examples of container languages include the SOAP or OAI-PMH formats. The containment function is defined within the rules of the SOAP language: a receiver of a SOAP XML instance interprets that document according to those containment rules, but the contained component is interpreted according to the rules of a second language. Similarly METS, although not only a container language, has well-defined components which do act as containers for other XML formats

At the other extreme are XML languages are intended for use within the context of other languages. For example, MathML can be used stand-alone, but is also intended to be embedded within other languages. Some XML languages can *only* be deployed in the context of another language e.g. languages like XLink or RDF/A provide only XML attributes, which are intended for use on the XML elements defined by another XML language. (Such languages are sometimes referred to as "parasite" languages as they require a "host".)

[Middle-ground examples e.g. RSS2.0/Atom, RDF/XML, DC in XML - they work because there is another data model layered on top of the XML Infoset]

2.7 Summary

- XML QNames (as representations of expanded names) are associated with components of XML documents (XML elements, XML attributes). Multiple XML elements in the same document may have the same expanded name.
- XML DTDs and XML Schemas describe structural constraints on a class of XML documents
- An XML language consists of an XML vocabulary (a set of names), a set of structural constraints (which may be expressed by a DTD or XML Schema), and a human-readable specification of how the named components are interpreted
- XML imposes no fixed meaning on the structural relationships between XML components (parent-child relationships between elements or owner relationships between attributes and elements), and the same relationship carries different meaning in different XML languages and in different contexts within the same XML language
- The same XML QName (representing an expanded name) may be used in the XML vocabulary of multiple XML languages; in each of those XML languages, the named component may be subject to different structural constraints
- Although there is some expectation that names are used consistently across XML languages, the meaning of a name is dependent on the XML language in which it occurs
- The interpretation of an XML document depends not only on the XML vocabulary, but on the rules of the XML language, and these rules - the "semantics" of the language - are available only to the human reader. In particular, they are not part of the XML Infoset, the XML "abstract model".

3. RDF, URI references, and RDF/XML

3.1 The RDF data model and URI references

The Resource Description Framework (RDF) set of specifications describe a means of constructing simple statements about resources.

Central to RDF is the idea of the resource, which can be anything you wish to describe - a document, a physical object, a person, an imaginary being, a concept, anything at all - and the idea of identifying resources using Uniform Resource Identifiers (URIs) (or more accurately URI references). In RDF, URI references are simply names for things. The fact that some URI references used in RDF may also be used by software applications to obtain access to digital objects is irrelevant to RDF. Also RDF treats URI references as "opaque" strings: the internal structure of a URI reference has no significance in RDF. It is important to note that an RDF application can not determine the relationship between a URI reference and a resource - it can only make use of the URI reference as a name.

(The nature of the relationship between URI references and resources has been part of the debate about "social meaning" in RDF. Essentially, URI references are used as if they always identify/denote a single resource, but that assumption is not part of the formal semantics of RDF. (I think I've got that right, but I may be oversimplifying.))

The basic building block of the RDF data model is the **triple**, consisting of a subject, a predicate and an object. The **subject** is a URI reference (or a "blank node"), the **predicate** is a URI reference, and the **object** is a URI reference, a blank node or a literal. (This document will not deal with "blank nodes" in any detail - for the purposes of the current discussion a blank node can be considered to be a sort of local identifier for a resource which is not identified by a URI reference.)

Each triple represents a statement: that statement asserts that a relationship exists between the two resources denoted by the subject and the object of the triple, and the type of that relationship is indicated by the predicate URI reference. A URI reference that is used as the predicate of a triple denotes a particular type of resource called a **property**.

As noted above, RDF does not deal with the relationship between a URI reference and the resource it denotes. Although this level of "meaning" - the difference between "having a title" and "having a subject", for example - may be used by the human interpreters of RDF statements, or by programmers writing software to operate on RDF data - it is not accessible to software. However, the RDF specifications, specifically *RDF Semantics* [RDFSEM], do provide a "formal meaning" for RDF and for the sets of URI references (vocabularies) defined by the RDF specifications. This "formal meaning" is defined in terms of the logical inferences that can be drawn, the "entailments" that follow, from the use of those URI references in RDF statements.

The following four triples represent four statements, each one stating a relationship between two resources:

Subject	Predicate	Object
http://example.org/doc/123	http://purl.org/dc/elements/1.1/creator	http://example.org/person/John
http://example.org/doc/456	http://purl.org/dc/elements/1.1/contributor	http://example.org/person/John
http://example.org/person/John	http://xmlns.com/foaf/0.1/name	"John Smith"
http://example.org/person/John	http://xmlns.com/foaf/0.1/knows	http://example.org/person/James

Example 12

Since RDF triples by definition accommodate only one subject and one object, a property describes a relationship between two resources, a binary relation. So a property is a "conceptual resource". It is still a resource, however, and a property URI reference can be the subject or object of an RDF triple, i.e. RDF allows you to create statements "about" a property in the same way as about other types of resource.

Subject	Predicate	Object
http://purl.org/dc/elements/1.1/creator	http://www.w3.org/2000/01/rdf-schema#label	"Creator"
http://purl.org/dc/elements/1.1/creator	http://www.w3.org/2000/01/rdf-schema#comment	"An entity primarily responsible for making the content of the resource."

Example 13

While the abstract model of an XML document is a tree, the abstract model for RDF is a "graph": a structure where "nodes" are linked together by "arcs". The subject and object of a triple are represented by nodes and the predicate is a labelled arc linking from the subject node to the object node. The triples in Example 12 would be represented as the following graph:

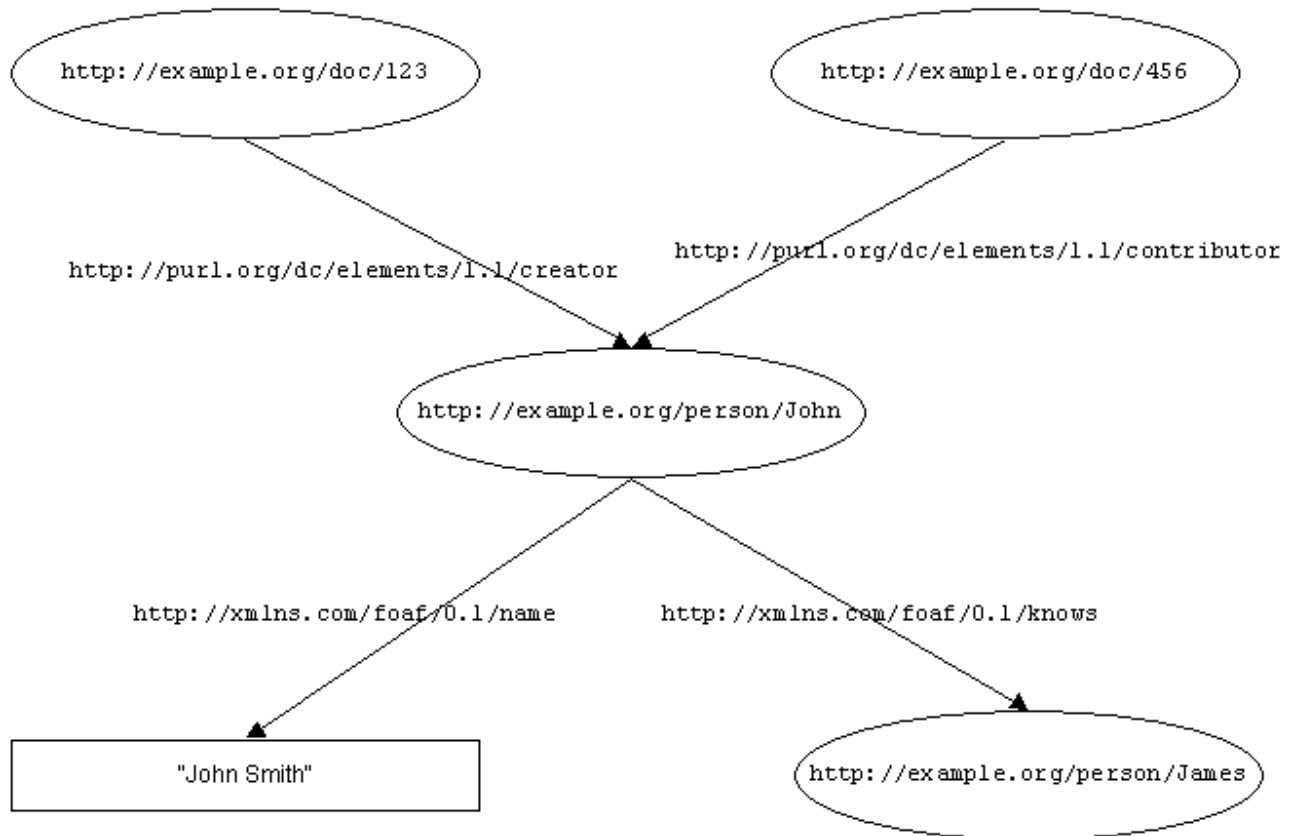


Figure 5: The RDF Graph

Just as the XML Infoset tree is an alternative view of an XML document, so the RDF graph is an alternative view of the subject-predicate-object triples.

In the RDF graph, the nodes are URI references that name resources of any type, and any node may be linked to an unlimited number of other nodes, and each of those links may carry any URI reference as a label. There is no order in an RDF graph.

The key difference between the XML Infoset tree and the RDF graph is that the RDF data model specifies that each node-arc-node triple is to be interpreted as a set of statement, whereas XML leaves it to each separate XML language specification to describe how the parent-child and attribute-element relationships in the tree are to be interpreted.

The triple/graph model also makes it easy to merge together two different graphs, two different sets of triples. The merged graph is simply the "union" of the two individual graphs, or the concatenation of the sets of triples, *but* with care taken to ensure that blank nodes (local identifiers) are maintained as distinct. This means that combining data from different sources, which is complex using XML, is relatively easy using RDF.

3.2 RDF Vocabularies

In the same way that XML does not provide a fixed set of XML element type names and attribute names, so RDF does not specify a fixed set of URI references that can be used in RDF triples. Rather RDF user communities deploy URI references that denote resources of interest to them. They need not only URI references to denote the particular resources (documents, books, images, concepts etc) they wish to describe, but also URI references to indicate the types of those resources and the properties used to describe their attributes and the relationships between them i.e. user communities define RDF **vocabularies** for their domains of interest.

The RDF Vocabulary Description Language (RDF Schema) [\[RDFS\]](#) provides....

(Something about classes and type-ing, subproperty/subclass)

(N.B. URIref opacity - tells you nothing about vocabulary etc.)

3.3 Syntaxes for serialising RDF

In order to exchange RDF data between applications, the data must be represented in some digital format. This process is referred to as **serialisation**. The RDF data model is independent of any specific serialisation syntax. In particular RDF does not rely on XML. There are several XML-based syntaxes for representing sets of RDF statements, and there are also several syntaxes for that are not based on XML.

3.3.1 RDF/XML

The RDF/XML specification [\[RDFXML\]](#) provides a set of rules for representing a set of RDF triples in XML. In the terms of the discussion of XML above, the RDF/XML

specification defines RDF/XML as an XML language.

The RDF/XML language specification defines a convention for representing RDF URI references as expanded names, encoded in documents as XML QNames. It is important to remember that there is a mapping taking place between XML QNames in the XML document and RDF URI references in the RDF graph, and that this is a convention specific to the RDF/XML language. It is *not* the case that the XML expanded name or the XML QName identifies the RDF property. And indeed a single URI reference may be expressed in RDF/XML using many different XML QNames. See [Example 14](#) and [Example 15](#) below.

Further, RDF/XML represents only some URI references as XML QNames (predicate URI references and URI references that represent the type of a resource: other URI references are encoded in full. (Also of course there are XML QNames used in RDF/XML that name components of the RDF/XML language but do not map to URI references (e.g. `rdf:Description`, `rdf:resource`, `rdf:parseType` etc).

The triples in [Example 12](#) could be represented in RDF/XML as follows. All of these XML documents are alternate representations/serialisations of the *same* RDF graph.

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:dc="http://purl.org/dc/elements/1.1/"
  xmlns:foaf="http://xmlns.com/foaf/0.1/">
  <rdf:Description rdf:about="http://example.org/doc/123">
    <dc:creator rdf:resource="http://example.org/person/John"/>
  </rdf:Description>
  <rdf:Description rdf:about="http://example.org/doc/456">
    <dc:contributor rdf:resource="http://example.org/person/John"/>
  </rdf:Description>
  <rdf:Description rdf:about="http://example.org/person/John">
    <foaf:name>John Smith</foaf:name>
    <foaf:knows rdf:resource="http://example.org/person/James"/>
  </rdf:Description>
</rdf:RDF>
```

Example 14

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:dc="http://purl.org/dc/elements/1.1/"
  xmlns:foaf="http://xmlns.com/foaf/0.1/">
  <rdf:Description rdf:about="http://example.org/doc/123">
    <dc:creator>
      <rdf:Description rdf:about="http://example.org/person/John" foaf:name="John Smith">
        <foaf:knows rdf:resource="http://example.org/person/James"/>
      </rdf:Description>
    </dc:creator>
  </rdf:Description>
  <rdf:Description rdf:about="http://example.org/doc/456">
    <dc:contributor rdf:resource="http://example.org/person/John"/>
  </rdf:Description>
</rdf:RDF>
```

Example 15

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:z="http://purl.org/dc/elements/1.1/c"
  xmlns:foaf="http://xmlns.com/foaf/0.1/">
  <rdf:Description rdf:about="http://example.org/doc/123">
    <z:reator>
      <rdf:Description rdf:about="http://example.org/person/John" foaf:name="John Smith">
        <foaf:knows rdf:resource="http://example.org/person/James"/>
      </rdf:Description>
    </z:reator>
  </rdf:Description>
  <rdf:Description rdf:about="http://example.org/doc/456">
    <z:ontributor rdf:resource="http://example.org/person/John"/>
  </rdf:Description>
</rdf:RDF>
```

Example 16

[Example 16](#) was designed to highlight that RDF/XML makes a mapping between XML expanded names and URI references. [Example 15](#) and [Example 16](#) have the same

XML document structure, a corresponding set of XML components. But those components have a different set of XML expanded names in each case. Yet they both represent the same RDF graph, the same set of RDF triples. It is important to remember that, to an RDF application, these variations in the serialisation syntax - which *would* be significant in an XML application - are quite invisible and have no significance: the QNames `dc:creator` (in [Example 14](#) and [Example 15](#)) and `z:creator` (in [Example 16](#)) are both simply a means of representing the single URI reference `http://purl.org/dc/elements/1.1/creator`, and many other prefix/namespace name/local name permutations are possible.

3.3.2 Other syntaxes

RDF/XML is just one syntax for the serialisation of RDF graphs. There are other XML-based syntaxes (e.g. TRIX,) and also text-based syntaxes not based on XML (e.g. N-Triples, Turtle etc). Many of these syntaxes incorporate a mechanism which permits the encoding of URI references using Qualified Names, though in the case of the non-XML syntaxes these conventions are unrelated to the concept of the **XML** Namespace. A single RDF application might read and write documents in many different RDF serialisation syntaxes, but all the different formats are representations of graphs, sets of triples.

For these reasons, it is important when discussing RDF - and particularly when comparing RDF and XML - to try to focus on the "abstract models" of the RDF graph and the XML tree. Comparison at the syntactic level may lead to confusion and false conclusions, particularly (as the examples above show) regarding the significance or otherwise of the *names* (QNames, expanded names) used to label XML components. This can be difficult at first for people accustomed to reading XML documents, but it is an absolutely vital step.

3.4 Summary

- URI references are identifiers for resources of any type.
- RDF provides a simple data model for making simple statements about resources in the form of triples composed of a subject, predicate and object. RDF statements make use of URI references as names.
- The RDF "abstract model" is that of a graph, with nodes linked together by labelled arcs.
- The RDF data model is defined quite independently of XML, but sets of RDF triples can be serialised as XML documents.
- (RDF Schema)

4. XML and RDF

4.1 Qualified Names in XML and RDF

In XML, XML QNames are used in XML documents to represent the XML expanded names (two part constructs made up of an XML Namespace Name and a local name) that form the vocabulary of an XML language. Those expanded names are used as the names of components in XML documents (XML elements, XML attributes). They are processed and interpreted according to the specification of that XML language. It must be emphasised that in XML generally, XML QNames are *not* URI references and they are not mapped to URI references.

In RDF, some text-based serialisation syntaxes provide a mechanism for using "qualified names" to abbreviate URI references. And in discussions of RDF generally, it is commonplace to find "qualified names" used, as abbreviations for those URI references, to refer to those properties and classes. So, for example, the property with the URI reference `http://purl.org/dc/elements/1.1/title` is sometimes referred to as `dc:title` or `DC.title`. The qualified name form is simply an abbreviation for the full URI reference.

In RDF/XML, URI references may be represented as XML expanded names, which are encoded as XML QNames used as XML element type names or XML attribute names. However, it is important to bear in mind that the XML components in XML documents are different things from the property itself, and that there is a mapping process taking place which is specific to this XML language.

A focus on the RDF/XML syntax to the exclusion of the RDF data model can lead to false assumptions about the use of names in XML languages and in RDF.

The vocabulary of an XML language (the set of expanded names which is encoded as QNames) is *not* the same thing as an RDF vocabulary (a set of URI references). And the existence of an XML vocabulary and the use of the corresponding QNames in XML documents does *not* result in the creation of a corresponding set of URI references. Approaching RDF on the basis that the QNames that have been used in an XML language can simply be redeployed in RDF/XML is not a coherent approach because it ignores the fact that in the two contexts the names apply to quite different entities and are interpreted in quite different ways.

Certainly, an XML QName currently used in any XML language (XHTML, MODS, METS etc.) *could* be deployed in an RDF/XML document. URI references do not have to be pre-declared before they appear in RDF triples. And depending on the context in which that XML QName is used in the RDF/XML document, an RDF/XML parser would generate a URI reference from the expanded name and present that URI reference in an RDF triple. It may appear as the predicate of an RDF triple, and on that basis an RDF application will infer that the generated URI reference denotes a property. But that property is not the same thing as the initial XML component

Consider a concrete example. The XHTML XML language includes an XML element type name `title` associated with the XML Namespace Name `http://www.w3.org/1999/xhtml`. That name can be deployed in an RDF/XML document:

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:xhtml="http://www.w3.org/1999/xhtml">
  <rdf:Description xhtml:title="DCMI Home Page" />
</rdf:RDF>
```

Example 17: RDF/XML

and an RDF/XML parser will generate the triple

Subject	Predicate	Object
(blank node)	<code>http://www.w3.org/1999/xhtml#title</code>	"DCMI Home Page"

Example 18

The RDF/XML parser generates a URI reference and infers that the URI reference denotes a property, but the XHTML specification does not provide any information

about a resource with the URI reference `http://www.w3.org/1999/xhtml:title` and there is no RDF Schema description of this property. The XHTML specification describes an XML language and describes only XML components with expanded names, to be interpreted in the context of an XML tree structure. The `xhtml:title` element is described as a container for a text string, not as a property.

Further, using the same XML name as the name of a different component in RDF/XML generates a *different* RDF graph:

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:xhtml="http://www.w3.org/1999/xhtml">
  <xhtml:title rdfs:label="DCMI Home Page" />
</rdf:RDF>
```

Example 19: RDF/XML

Subject	Predicate	Object
(blank node)	<code>http://www.w3.org/1999/02/22-rdf-syntax-ns#</code>	<code>http://www.w3.org/1999/xhtml:title</code>
(blank node)	<code>http://www.w3.org/2000/01/rdf-schema#label</code>	"DCMI Home Page"

Example 20

Here the rules of the RDF/XML language dictate that `xhtml:title` is interpreted as representing a URI reference that provides the type of the resource, and an RDF processor infers that that URI reference denotes a class. Again no such interpretation is covered by the XHTML specification.

And consider the rest of the vocabulary of the XHTML language. Any name from that vocabulary (`xhtml:html`, `xhtml:p`, `xhtml:em`) could be deployed in the same way, and an RDF triple generated, but such triples do not form - or do not necessarily form - a coherent representation of the information conveyed by the XHTML XML language.

Such an approach is simply transposing names from one context to another with no consideration for the contexts within which the names are deployed and interpreted. In short the names used in an XML language can not simply be transposed into RDF (or rather into RDF/XML), or at least not in any meaningful way. (See [Note 4](#))

To map between XML and RDF - or rather, as discussed in the next section, between an XML language and the RDF data model - , it is necessary to consider not simply the vocabulary of the XML language but the meaning that the XML language is intended to convey, the semantics of the language that are not accessible from the XML Infoset.

Note that it may emerge that that "semantic" analysis/re-modelling/mapping process *does* lead to a decision to use URI references that are encoded using QName forms that are similar to those used in the XML language e.g. in the example above the mapping might specify that an RDFS class called `http://www.w3.org/1999/xhtml:title` is required. But that decision would then be the result of the considered analysis and re-modelling, taking into account the contexts of the XML language and the RDF data model, rather than a "blind" transfer of the names.

4.2 Mapping XML Languages to the RDF data model

It may well be the case that an XML document *does* represent simple statements about resources. But there is nothing in the XML specification that describes how to represent such statements in the XML tree structure. Different XML language designers make different decisions about how the document author should encode this information in an XML document. As a result, the recipient of the XML document needs access to the specification of the XML language if they are to interpret the XML document as anything other than a simple tree structure.

As a consequence, there is no single way of interpreting an XML tree structure in terms of the RDF data model. Rather it is necessary to:

- analyse the specification of the individual XML language to determine what information is being represented by that language, the semantics of the XML language that are not directly accessible from the XML Infoset
- develop a specification of how that information should be represented using the RDF data model (using URI references from existing RDF vocabularies, or by developing new RDF vocabularies for the concepts specific to the application domain within which the XML language is used, or by a combination of the two approaches)
- describe the mapping or correspondence between constructs used in the XML language and statements made using the URI references selected or created

This is a re-modelling and mapping process: the names and components used in XML documents are quite different from those used in RDF graphs. It is also an XML-language-specific process because, as described above, the interpretation of names and components varies across XML languages. It may also vary according to the context of the named component within the same XML language. Because of the nature of, and the differences between, the XML and RDF data models, there may be no simple one-to-one correspondence between XML element type names and XML attribute names on the one hand and RDF URI references on the other.

Depending on the design of the XML language, there may be regular "patterns" used in that language which make this mapping process easier, and encouraging the adoption of such patterns may facilitate the development of the mapping. But in the general case there is no one set of rules that can be applied to all existing XML languages. This was summarised concisely in a [recent message](#) to the W3C RDF Interest Group mailing list

There is no default mapping of XML document instances to RDF triples, other than the representation of the infoset in RDF, since XML is a generic framework that allows people to create an unbounded amount of applications on top of it.

- Sean B. Palmer, 2005-01-15

Note: None of the above is intended to suggest that RDF is better or worse than XML, simply that they are different and those differences must not be ignored. Both XML and RDF have their uses, and indeed there are many cases where XML may be a better choice than RDF (e.g. when data deals with N-ary relations: while it may be possible to re-model it as a set of binary relations, that re-modelling may not be efficient.)

5. Dublin Core and Dublin Core Application Profiles

5.1 The DCMI Abstract Model

The DCMI Abstract Model defines a DC metadata **description** as a set of **statements** about a single subject **resource**. Each statement is made up of:

- a reference to a **property**, in the form of a URI reference (a **property URI reference**)
- a reference to a second resource, the **value**, in the form of either a URI reference (a value URI reference), a **representation** of the value or a **description** of the value

A statement may also contain a reference to a **vocabulary encoding scheme** and a **syntax encoding scheme**, again both in the form of URI references. DC metadata descriptions are typically grouped together as **description sets**.

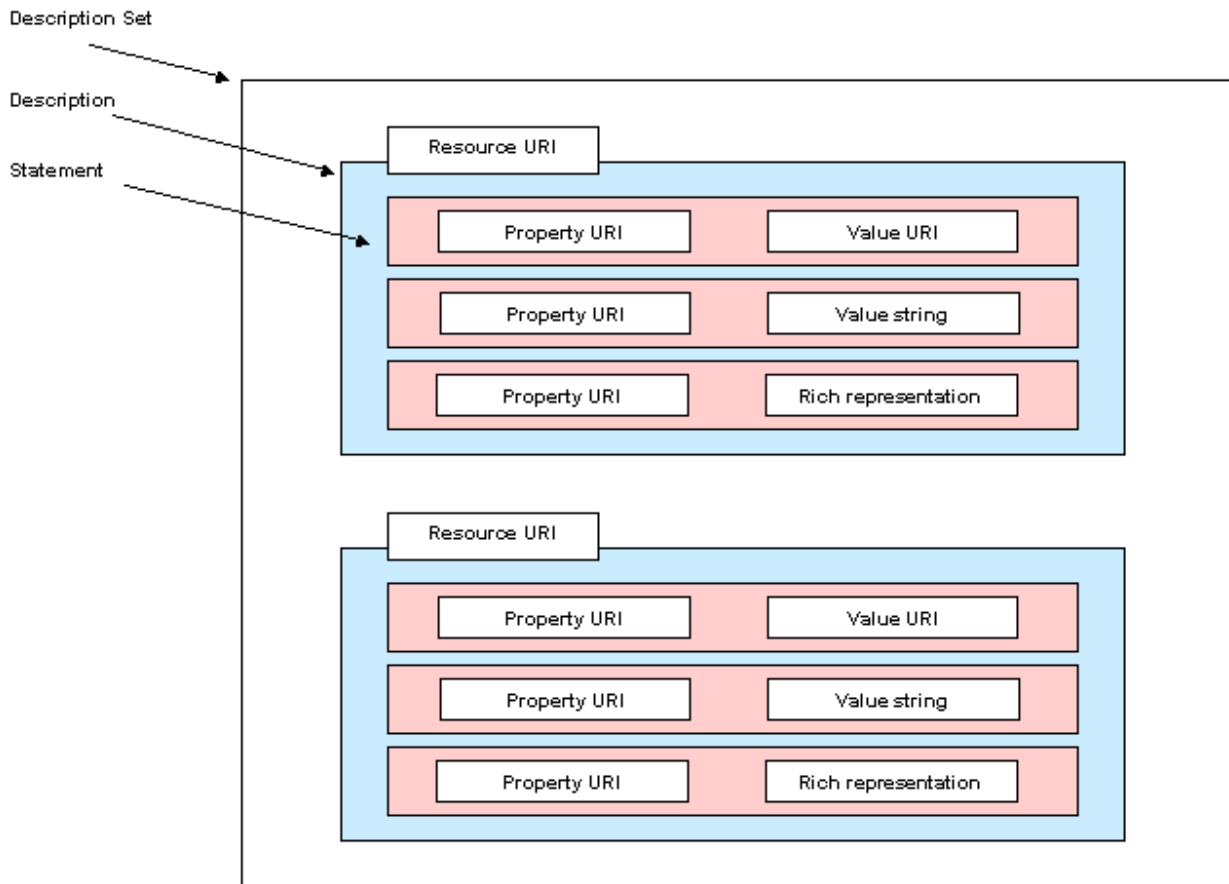


Figure 6: A DC Metadata Description Set

Properties and encoding schemes are referred to in DC metadata descriptions by means of URI references. Without a URI reference, a property or encoding scheme can not be referred to in a DC metadata description.

The Abstract Model is essentially a variation on the RDF data model. Although DC statements appear to have two parts, they are always associated with a description, and a description applies to exactly one resource, so DC statements are really triples.

The Abstract Model differs in

- its treatment of literals
- the introduction of description and description set

But essentially all the comments made above about RDF - and particularly the distinctions between RDF and XML - apply to the DC Abstract Model. The Abstract Model is not based on the XML tree model.

5.2 Dublin Core Terms, "Elements", Qualified Names and URI references

DCMI assigns URI references to all the "terms" it defines following the policies and conventions described in the *Namespace Policy for DCMI Terms* document [DCMINS]. Those URI references take the form of PURLs, e.g. <http://purl.org/dc/elements/1.1/title>.

Note that DCMI uses the word "term" to refer to the conceptual resource rather than the URI reference defined to it.

The "terms" defined by DCMI are of three types:

- elements and element refinements
- encoding schemes
- "terms from a controlled vocabulary"

The nature of these "terms" is described by the DCMI Abstract Model. A DC element or element refinement is a property: "a specific aspect, characteristic, attribute, or relation" that can be applied to the description of a resource. An encoding scheme is a class. Although the Abstract Model distinguishes between vocabulary encoding schemes and syntax encoding schemes, DCMI currently models all encoding schemes as classes.

Although DCMI documentation refers to "terms from a controlled vocabulary", the only controlled vocabulary that it currently maintains is the DCMI Type Vocabulary, and the "terms" in this vocabulary have the specific characteristic that they are all classes. This would not necessarily be the case for other "controlled vocabularies", where the "terms" may be resources of any type.

A DC element is **not** the same type of thing as an XML element (or element type):

- an **XML element** is a component within an XML document and its characteristics are defined by the XML and XML Infoset specifications. XML elements have element content, child elements, and attributes; an XML element has an XML element type name, which takes the form of an expanded name; and the interpretation of an XML element depends on its use in the context of an XML language, which is described by the human-readable specification of that language
- a **DC element** is a property and its characteristics are defined by the RDF specifications and the DCMI Abstract Model; a property is identified by a URI reference, and that URI reference can be used as a predicate in an RDF triple or a property URI in a DC metadata statement

Although all the DC "terms" - properties and classes - are identified by URI references, it is common in DCMI documentation to find "qualified names" used, as abbreviations for those URI references, to refer to those properties and classes. So, for example, the property with the URI reference <http://purl.org/dc/elements/1.1/title> is sometimes referred to as `dc:title` or `DC.title`. Some text-based syntaxes for serialising RDF graphs also support this construction. The qualified name form is simply an abbreviation for the full URI reference.

However, DC metadata descriptions may also be serialised as XML documents, either using the RDF/XML language or the DC-XML language. In both those XML languages, URI references may be represented as XML expanded names, which are encoded as XML QNames used as XML element type names or XML attribute names. However, it is important to bear in mind that the XML components in XML documents are different things from the property itself, and that there is a mapping process taking place which is specific to these XML languages.

Usually, it is possible to establish from the context whether a qualified name is being used as a shorthand for a URI reference or as an encoding of an XML expanded name, but care needs to be taken.

Taken together, however, these two factors - the unqualified use of the word "element" to refer to two different things and the use of qualified names in two different contexts - have contributed to some confusion. It must be emphasised that in XML generally, XML QNames are *not* URI references and they are not mapped to URI references: an XML QName is used to encode an expanded name in an XML document, and an expanded name is a two part construct, made up of an XML Namespace Name and a local name. It is interpreted in the context of the XML language in which it is used. (See [Note 5](#))

Because the DCMI Abstract Model is based on or similar to the RDF data model, all the points made about XML and RDF in [section 4](#) above apply to XML languages and DC metadata applications. The names and components used in an XML language can not be deployed in a DC metadata description: rather it is necessary to follow the process of analysing the meaning that the XML language (or some subset of constructs within the XML language) is intended to convey and developing the *new* set of "terms" required - and that set of "terms" may include properties, classes, and other resources depending on what information is to be represented.

5.3 Dublin Core Application Profiles and XML Languages

As noted in the introduction, DCMI has not defined a formal model for what a Dublin Core Application Profile (DCAP) actually is (See [Note 6](#)). Probably the closest to such a model that DCMI has at the present is the statement in the CEN CWA 14855 that:

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.

However CEN CWA 14855 is not clear about many aspects of a DCAP. In particular, the suggestion that "terms" may be referred to within a DCAP even if they are not identified by URI references has caused confusion.

With reference to the the Abstract Model, it seems reasonable to consider that a DCAP specifies the "terms" that are referenced within a particular class of descriptions or description sets. As discussed in [section 5.1](#) these "terms" are **properties** and **classes**. So the "terms" referenced or "used" in a DCAP are also properties and classes. A DCAP specifies the properties that are used to describe particular types of resource (classes), and how those properties are deployed, including any constraints on their values i.e. any classes to be used as encoding schemes. The properties and classes are referenced by citing their URI references, which may be drawn from any RDF vocabularies, including vocabularies developed by agents other than DCMI.

Central to the idea of the DCAP is the idea that the DCAP does not itself declare new "terms", but rather references or "uses" (or reuses) "terms" that are declared elsewhere. The widespread adoption of XML has led to suggestions that the components used in XML languages are terms that can be referenced in DCAPs.

However, as discussed in detail in the previous section the names and components used in an XML language can not be deployed in an RDF graph or DC metadata description (except as XML Literals or rich representations), and since the very purpose of a DCAP is to specify the URI references that can occur in a DC metadata description, it follows that it is not meaningful to reference them in a DCAP. A DCAP can not "reuse XML elements".

If it is required for a DCAP to describe how to express some information that is currently expressed in an XML language, then it is necessary to develop a means of representing that same information within the framework of the DCMI Abstract Model and the RDF data model i.e. to establish a means of expressing the information that is currently represented using components within an XML tree structure in terms of the statement-based models of the DCAM and RDF.

That process is outlined in section 4 above. It involves either establishing how that information can be represented using an existing RDF vocabulary or developing a **new** RDF vocabulary i.e. identifying the set of properties, classes, and other resources required to express that information in the statement-based model, and providing URI references so that they can be referenced by DC metadata descriptions. Those new properties, classes and URI references are different things from the names and components used in the XML documents, and there may be no simple one-to-one correspondence between the names of components in the XML language and the URI references of the RDF vocabulary.

In order for those new terms to be useful to consumers of the data, and to be reused by the authors of other data, it is useful to provide descriptions of what those newly-coined URI references denote, either in the form of human-readable documentation, or machine-processable descriptions made using the RDF Schema language, or both.

If the entire XML language or some subset of the constructs used within an XML language is mapped into the DC/RDF data models, then documentation on that

mapping and/or tools that apply the mapping to XML documents (e.g. XSLT transforms) are valuable to other implementers. See also GRDDL [[GRDDL](#)].

The naming and ownership of the URI references of the new RDF vocabulary is a social-political question rather than a technical one, and it is important to distinguish this issue from the semantic modelling/mapping issue. To RDF, URI references are simply opaque strings. As the discussion in section 4 highlights, the properties and classes are different things from the components of an XML language and it is unlikely that there will be a simple one-to-one correspondence between them i.e. there will probably not be a simple correspondence between expanded names/QNames in the XML language and URI references in the RDF vocabulary. It may also be the case that the mapping and the RDF vocabulary is developed quite separately from the XML language, even by an agency that is not the owner of the XML language.

Whatever names are used within the XML language and the RDF vocabulary, it will be necessary to describe a *mapping between* the XML language and the RDF model and to be absolutely clear that the names (URI references) in the RDF vocabulary are different from the names (expanded names/QNames) in the XML vocabulary (see section 4.1).

It *may* be possible to select the URI references of the new RDF vocabulary so that when they are represented in RDF/XML as expanded names encoded as XML QNames, those names correspond to the expanded names and QNames used in the initial XML vocabulary. (But see [Note 4](#): if a URI reference has been assigned to an XML element type, then that same URI reference can *not* also be assigned to an RDF property. However, bearing in mind a mapping is always required, and the content models for the named components in RDF/XML will be *different* from the content models of any component that uses that same name in the XML language - e.g. in the latter a sub-tree structure may be available - it may be preferable to ensure that URI references are chosen so that their XML expanded name/QName representation does *not* duplicate any of the names used in the vocabulary of the XML language. The LOM XML binding and the LOM RDF binding take this latter approach. There is *no* overlap between the expanded names/QNames used in the LOM XML binding and those used when a LOM RDF graph is serialised in RDF/XML. It is clear to the user that they are quite different XML vocabularies used in the context of two different XML languages (LOM XML and RDF/XML). That has no impact on the capacity to describe the mapping between the LOM XML language and the RDF data model.

However, the human users and owners of the two vocabularies may feel it is appropriate that the names in the two vocabularies carry some indication of a common source, e.g. that the URI references used in the RDF vocabulary are in some way similar to the URI references used as XML Namespace Names in the XML language.

5.4 Summary

- a DCAP specifies the "terms" that are referenced within a particular class of DC metadata description sets, and these "terms" are properties and classes
- a DCAP does not itself declare new "terms", but rather references or "uses" (or reuses) "terms" that are declared elsewhere
- the names and components defined by an XML language can not be deployed in a DC metadata description and so can not be referenced in a DCAP
- to express in a DC metadata description some information that is currently expressed in an XML language, it is necessary to carry out an analysis of the semantics of that XML language and to develop a means of representing that same information within the framework of the DCMI Abstract Model and the RDF data model, and to specify an RDF vocabulary to do that (either referencing an existing vocabulary or developing a new one)
- a new RDF vocabulary (if required) should be described in human-readable documentation and using the RDF Schema language
- the naming and ownership of any new RDF vocabulary is a separate issue from the semantic analysis/modelling/mapping issue. While it may be socially/politically advantageous that names in XML vocabulary and names in an RDF vocabulary carry some indication of common source/ownership, care must be taken to make it clear that these are different sets of names used in the context of different specifications, and there is a mapping between the XML language and the RDF/DC data model(s).

6. Conclusions/Recommendations

There is a good deal of confusion surrounding the concept of the DCAP and their construction. The absence of a clear specification of what a DCAP is, together with misunderstandings about XML and RDF, and some ambiguity in DCMI's use of terminology (or users' interpretation of that terminology), have meant that although the general idea of the DCAP has been widely embraced, in practice it has been interpreted and implemented in different ways, sometimes significantly different ways. Sometimes those interpretations and implementations are not consistent with the DCMI Abstract Model and/or with the data models used in the XML and RDF specifications.

This document has sought to highlight a small subset of the issues, particularly on the problem of "reusing" or "mixing and matching" "terms". It has tried to clarify in some detail why an unqualified notion of "reuse" is problematic, with particular reference to XML and (in sections 5.3 and 5.4) to suggest how those problems might be addressed so that the conditions can be put in place to make the promise of "mixing and matching" realisable.

Notes

[1] This is a slight simplification, since XML documents can also contain other types of item such as comments and processing instructions, and these can form part of XML element content, but for the purposes of this document, we consider XML documents to be made up of XML elements and attributes.

[2] Again this is a simplification as not only are there other types of information item (see Note 1), but the InfoSet provides one "Character" information item for each character of element content: here a sequence of characters is represented as a single item. The Infoset also provides items related to the use of XML Namespaces.

[3] This is typically a human-readable document, though specifications like GRDDL [[GRDDL](#)] represent an attempt to disclose at least some of that information in machine processable form, by providing access to an XSLT transform (specific to that XML language) that generates an RDF/XML document from the XML document.

[4] A fragment of XML conforming to *any* XML language could, of course, be used as an RDF XML Literal (or a "rich representation" in the terms of the DCMI Abstract Model. In that case it is not interpreted by the RDF/XML parser; it is simply passed to the application as an XML fragment.

[5] The "CORES Resolution" [[CORESRES](#)] encouraged the owners of metadata standards to assign URI references to their "elements", the "units of meaning comparable and mappable to elements of other standards", but it did not specify what "comparable and mappable" meant. As a consequence the owners of different standards assigned URI references to "elements" that are created within different frameworks and rely on those frameworks for their meaning and interpretation. The assignment of a URI reference to an "element" means that it can be unambiguously cited - and it could be the subject of a DC metadata description - but it does not change the nature of the "element": and it does not mean that it is meaningful to use that URI reference as, e.g., a property URI in a DC metadata description. Indeed saying that a single URI reference denoted *both* an element defined within a hierarchical model *and* a property would contradict the principle that a URI should identify a single resource.

[6] I can't emphasize strongly enough how problematic it is that DCMI has no formal definition of what a DCAP actually is, and that documents like CEN CWA 14855 present a rather "loose" specification. I propose a notion of a DCAP here that seems consistent with most of the approaches to the idea that I've seen and which is based on the DC Abstract Model. But I readily admit I'm influenced by my own experience with various projects, and it is just one possible model for a DCAP!!! Someone else could propose a quite different, but equally valid, notion of a DCAP, also based on the Abstract Model (e.g. that a DCAP defined an application-specific XML language for representing DC metadata descriptions), and the arguments I make below would have to be modified for that case. (As an aside, I think a better name for what I describe here would be a DC Description Set Profile!)

References

[DCAPUB]

Thomas Baker, *DCMI Usage Board Review of Application Profiles*
<http://dublincore.org/usage/documents/profiles/>

[CWA14855]

CEN CWA14855 - *Dublin Core Application Profile guidelines*
<http://www.cenorm.be/iss/cwa14855/>

[DCMIAM]

DCMI Abstract Model
<http://dublincore.org/documents/abstract-model/>

[XML]

Extensible Markup Language (XML) 1.0 (Third Edition). W3C Recommendation 04 February 2004.
<http://www.w3.org/TR/REC-xml>

[XML]

XML Schema Part 0: Primer Second Edition. W3C Recommendation 28 October 2004.
<http://www.w3.org/TR/xmlschema-0/>

[XMLNS]

Namespaces in XML. W3C Recommendation 14 January 1999.
<http://www.w3.org/TR/REC-xml-names>

[XMLNS1.1]

Namespaces in XML 1.1. W3C Recommendation 04 February 2004.
<http://www.w3.org/TR/xml-names11>

[XMLINFO]

XML Information Set (Second Edition). W3C Recommendation 04 February 2004.
<http://www.w3.org/TR/xml-infoset>

[XPATH]

XML Path Language (XPath) Version 1.0. W3C Recommendation 16 November 1999.
<http://www.w3.org/TR/xpath>

[XQUERY]

XQuery 1.0: An XML Query Language. W3C Working Draft 11 February 2005.
<http://www.w3.org/TR/xquery/>

[XMLNS1.1]

[*Editorial Draft*] *Versioning XML Languages*. Proposed TAG Finding 16 November 2003.
<http://www.w3.org/2001/tag/doc/versioning.html>

[to be confirmed]

[DCXML]

Guidelines for implementing Dublin Core in XML
<http://dublincore.org/documents/dc-xml-guidelines/>

[RDFCAS]

RDF/XML Syntax Specification (Revised) W3C Recommendation 10 February 2004.
<http://www.w3.org/TR/rdf-syntax-grammar/>

[RDFCAS]

Resource Description Framework (RDF): Concepts and Abstract Syntax W3C Recommendation 10 February 2004.
<http://www.w3.org/TR/rdf-concepts/>

[RDFSEM]

RDF Semantics W3C Recommendation 10 February 2004.
<http://www.w3.org/TR/rdf-mt/>

[RDFS]

RDF Vocabulary Description Language 1.0 (RDF Schema) W3C Recommendation 10 February 2004
<http://www.w3.org/TR/rdf-schema/>

[DCMINS]

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

[GRDDL]

Gleaning Resource Descriptions from Dialects of Languages (GRDDL) W3C Coordination Group Note 13 April 2004
<http://www.w3.org/TR/rdf-schema/>

[CORESRES]

CORES Standards Interoperability Forum Resolution on Metadata Element Identifiers
<http://www.cores-eu.net/interoperability/cores-resolution/>

Valid XHTML 1.0!

Valid CSS!



[Home](#) > [Documents](#) > [Dc-elem-refine](#) >

Element Refinement in Dublin Core Metadata

Creator: Pete Johnston
Date Issued: 2005-06-13
Identifier: <http://dublincore.org/documents/2005/06/13/dc-elem-refine/>
Replaces: <http://dublincore.org/documents/2005/04/11/dc-elem-refine/>
Is Replaced By: Not applicable
Latest Version: <http://dublincore.org/documents/dc-elem-refine/>
Status of Document: This is a DCMI [Recommended Resource](#).
Description: This document describes the concept of "element refinement" as used in Dublin Core metadata. It seeks to explain the consequences of stating that one property "refines" a second property. The purpose is to clarify that in some cases it may be appropriate and useful to make such an assertion and in other cases such an assertion may result in contradictions.

Elements, Properties and Statements

Dublin Core elements and element refinements are properties. A property is "a specific aspect, characteristic, attribute, or relation used to describe resources". According to the *DCMI Abstract Model*, all these "aspects, characteristics, attributes and relations" involve relationships between resources [[DCMIAM](#)]. Each property corresponds to a type of relationship, such as the notion that a resource "is created by" an agent (the agent is a second resource), or that a resource "is about the subject of" some concept (the concept is also a resource).

A property is itself a resource, a "conceptual" resource. When DCMI adds a property to one of its "DCMI Namespaces", it creates a human-readable description of that concept, and it assigns a globally unique identifier to the property, in the form of a URI.

The scope of URIs is global: the URI is used as if it denotes that same concept, that same relationship type, wherever it is cited. Further, the persistence policies described in the *Namespace Policy for the Dublin Core* guarantee that the URIs DCMI assigns to its properties will always denote that same essential concept [[DCMINS](#)]. So, the assignment of a URI means that other parties can use this unique identifier, and the combination of its global uniqueness and persistence mean that the reference is unambiguous.

The URI assigned to the property can be used in statements in Dublin Core metadata descriptions. According to the *DCMI Abstract Model*, a DC metadata description is a set of one or more statements about a single resource, and a statement is a two-part construct consisting of a reference to a property and a reference to a second resource, a value [[DCMIAM](#)].

The reference to the subject of the description is made in one of two ways. The description may use a URI to reference the subject resource explicitly (the "resource URI"). Alternately the subject of the description may be implicit, depending on the format and/or the context in which the description occurs: for example, for a DC metadata description embedded in an XHTML document [[DCHTML](#)], that XHTML document is the subject of the description. For simplicity, the examples presented in this document reflect the case where a resource URI is provided.

The reference to a property also takes the form of a URI (the "property URI"). The reference to the value may take

the form of a URI or a "value representation". For the purposes of this discussion, the examples show the simplest cases where that reference takes the form of a URI (a "value URI") or a representation in the form of a string (a "value string").

Each statement asserts that a relationship of the type indicated by the property exists between two resources: the resource that is the subject of the description, and the value (see note [1]):

Resource URI	Statements	
ex:book1	Property URI	Value URI
	dc:subject	ex:SemanticWeb

Element Refinement

In addition to providing a definition and identifier for each of the properties it declares, DCMI also describes relationships between these properties. If the definitions of two properties are such that whenever two resources are related by the first property they are also related by the second property, the first property is said to "refine", or to be a "refinement" of, the second property.

So for example, the definition of the property `dcterms:created` is "Date of creation of the resource", and the definition of the property `dc:date` is "A date associated with an event in the life cycle of the resource". The date of creation of a resource is always "a date associated with an event in the lifecycle of the resource", so the `dcterms:created` property refines the `dc:date` property.

The machine-processable schemas published by DCMI include descriptions of all DC elements and element refinements. In the description of an element refinement, a statement is included using the property URI `rdfs:subPropertyOf` from the RDF Vocabulary Description Language (RDF Schema) [RDFS]. This states that a relationship exists between two properties, and the nature of that relationship is defined by the RDFS concept `rdfs:subPropertyOf`:

The property `rdfs:subPropertyOf` is an instance of `rdf:Property` that is used to state that all resources related by one property are also related by another [RDFS].

This `rdfs:subPropertyOf` assertion enables a human reader or a software application to infer new information when they encounter a statement made using the "refinement" or subproperty (see note [2]).

So if it is asserted that

```
dcterms:created rdfs:subPropertyOf dc:date
```

and a statement is made using `dcterms:created` as a property URI, e.g.

Resource URI	Statements	
ex:book1	Property URI	Value String
	dcterms:created	"1973-05-05"

then it can be inferred that it is also true that

Resource URI	Statements	
ex:book1	Property URI	Value String
	dc:date	"1973-05-05"

This outcome holds for **all** statements made using the URI of the element refinement as a property URI: whenever

two resources are related by the element refinement, they are also related by the corresponding element. So an assertion that one element refines another - that an `rdfs:subPropertyOf` relationship exists between the properties - should be made only when the definitions of the properties support that conclusion.

"To refine or not to refine"

If there is just one case where the inferred statement would not be true, then the refinement/subPropertyOf relationship should not be asserted.

Consider, for example, the case of the properties `marcrel:OWN` ("The person or organization that currently owns an item or collection") and `dc:contributor` ("An entity responsible for making contributions to the content of the resource"). Both properties describe relationship types that relate a resource to an "entity", an agent capable of some action. (The `marcrel:OWN` property is part of a set of properties defined by the Library of Congress, based on the MARC Relator Codes [[MARCREL](#)].)

And for a specific resource, it may well be true that a single entity is both an owner of and a contributor to that resource. But that does not apply in **all** cases. i.e. there are some resources where the entity who is the owner of the resource has not made a contribution to the content of the resource: not all resource owners are resource contributors. If `marcrel:OWN` was described as a refinement of `dc:contributor`, then that would mean that **every** statement using `marcrel:OWN` as a property URI would result in a statement using `dc:contributor` as a property URI, which would not be desirable.

Note also that the **absence** of a subproperty assertion in no way limits the capacity of the metadata author to say that, **for any given resource**, the same entity is both the owner and the contributor. The metadata author simply makes the two statements separately:

Resource URI	Statements	
ex:book1	Property URI	Value URI
	dc:contributor	ex:agent1
	marcrel:OWN	ex:agent1

As a second example, consider the case of `dc:date`, defined as "A date associated with an event in the life cycle of the resource". If an implementer uses a property `exterms:updatingFrequency` to indicate "The periodicity of modifications to the resource", and describes that property as an element refinement of `dc:date`, then statements such as the following might be inferred:

Resource URI	Statements	
ex:document1	Property URI	Value String
	dc:date	"Monthly"

Resources which are appropriate values for statements using `exterms:updatingFrequency` are not appropriate values for statements using `dc:date`, so it is **not** appropriate to describe that property as an element refinement of `dc:date`.

Similarly, consider `dc:rights`, defined as "Information about rights held in and over the resource". Suppose an implementer uses a property `exterms:privacyIndicator` to indicate whether a document should be publicly available or not, and specifies that Boolean (yes/no) values should be used. If that property is described as an element refinement of `dc:rights`, that would result in statements such as the following being inferred:

Resource URI	Statements	
ex:document1	Property URI	Value String
	dc:rights	"Yes"

There may be an argument that strictly speaking a Boolean value does not contradict the definition of `dc:rights`, but it would be difficult to consider the value "Yes" to be "information about rights held in and over the resource".

So, again, it is **not** appropriate to describe `exterms:privacyIndicator` as an element refinement of `dc:rights`, because of the statements that would be inferred.

"How many?" : Multiple Refinement Relationships

In the declarations that DCMI makes, any given property is the subject of only one `rdfs:subPropertyOf` relationship: a DC element refinement refines exactly one element - though an element may be refined by multiple element refinements.

However, in principle, multiple assertions might be made, with the result that when the property is used in a statement **multiple** additional relationships can be inferred to exist.

So if it is asserted that

```
exterms:bookDistributor rdfs:subPropertyOf exterm:s:distributor
```

and also

```
exterms:bookDistributor rdfs:subPropertyOf dc:publisher
```

and a statement is made using `exterms:bookDistributor` as a property URI, e.g.

Resource URI	Statements	
ex:book1	Property URI	Value URI
	exterms:bookDistributor	ex:Company1

then the following **two** statements are also true:

Resource URI	Statements	
ex:book1	Property URI	Value URI
	exterms:distributor	ex:Company1
	dc:publisher	ex:Company1

Note that it is **not** a question of choosing one option over the other, or two applications behaving in different ways: **both** statements are implied in **all** cases, and as long as the two applications have "knowledge" of the two subproperty relations, they should both generate the same inferences.

"Who says so?": DCMI Namespaces and other Metadata Vocabularies

The capacity to assert the existence of `rdfs:subPropertyOf` relationships involving properties from the DCMI Namespaces is not limited to DCMI.

The publisher of another vocabulary may wish to declare that a property in that vocabulary is a subproperty of a property from the DCMI Namespaces (or even that a property from the DCMI Namespaces is a subproperty of a property from their vocabulary).

The Library of Congress defines a set of properties based on the MARC Relator Codes [[MARCREL](#)], that can be used to assert relationships between resources and agents. It is useful that, where appropriate, subproperty relations between these properties and properties from the DCMI Namespaces are declared. e.g.

```
marcrel:ARR rdfs:subPropertyOf dc:contributor
```

(`marc:rel:ARR` denotes a property which links a musical composition and its arranger.)

With such information available, a Dublin Core application that encounters

Resource URI	Statements	
ex:music1	Property URI	Value URI
	marc:rel:ARR	ex:person1

can derive the statement

Resource URI	Statements	
ex:music1	Property URI	Value URI
	dc:contributor	ex:person1

This means that an application that has no "prior knowledge" of `marc:rel:ARR`, but which does derive the appropriate inferences from the assertion of the `rdfs:subPropertyOf` relationship, can make use of the `dc:contributor` statement. Such a process enhances interoperability between metadata applications.

A subproperty assertion may be made by a third party who is not the owner/publisher of either of the properties involved. Suppose a metadata vocabulary has been constructed by a designer with no knowledge of the existence of Dublin Core. In their descriptions of their properties they have made no references to the notion that their property `exterm:songwriter` is related to `dc:creator`. If an implementer is working with both that metadata vocabulary and with the DCMI vocabularies, it may be useful for them to make a subproperty assertion:

```
exterm:songwriter rdfs:subPropertyOf dc:creator
```

so that, when their Dublin Core application encounters

Resource URI	Statements	
ex:music1	Property URI	Value URI
	exterm:songwriter	ex:person1

that application can derive the statement

Resource URI	Statements	
ex:music1	Property URI	Value URI
	dc:creator	ex:person1

Finally, it should be re-emphasised that the existence of a subproperty relationship with a DC property is not a requirement for the use of properties from other metadata vocabularies in DC metadata. In the case of the MARC Relator properties, a subproperty relation may exist in many cases, but those properties for which no such relation exists, such as the owner property `marc:rel:OWN`, can still be deployed in statements in DC metadata descriptions.

Summary

- A declaration that one property refines, or is a subproperty of, a second property is an assertion of a **specific type of relationship** between the two properties.
- The meaning of the refinement/subproperty relationship is defined by the RDF Vocabulary Description Language (RDF Schema): the existence of a subproperty assertion states that **all resources related by one property are also related by the second property**.
- A subproperty assertion is **global**: it applies to **all** occurrences of the property URI in a statement; care should be taken not to assert subproperty relations where the inferences that may be drawn are incorrect

- or contradictory.
- Refinement/subproperty relations may be asserted to exist between **any** two properties.
 - A property may be the subject of **multiple** refinement/subproperty relations.
 - The existence of subproperty assertions about a property enables an application to infer **additional statements** from statements made using that property; the availability of subproperty assertions supports semantic interoperability between applications
 - A subproperty assertion is **not a requirement** for using a property from another vocabulary in Dublin Core metadata descriptions.
 - Subproperty assertions may be made by the owner(s) of the properties or by a third party.

Notes

[1] For the sake of brevity, in the examples, URIs are represented by Qualified Names. Prefixes are assumed to be associated with Namespace Names as follows:

- **dc:** <http://purl.org/dc/elements/1.1/>
- **dcterms:** <http://purl.org/dc/terms/>
- **dcmitype:** <http://purl.org/dc/dcmitype/>
- **marcrel:** <http://www.loc.gov/marc/relators/>
- **rdf:** <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
- **rdfs:** <http://www.w3.org/2000/01/rdf-schema#>
- **exterms:** <http://example.org/terms/>
- **ex:** <http://example.org/things/>

N.B. At the time of writing, the URIs to be assigned to the MARC Relator properties are still under discussion. The URIs used here are provisional, and should not be cited without first checking the Library of Congress Web site for the authoritative identifiers of the properties.

[2] This document does not concern itself with the details of **how** an application obtains information about the existence of subproperty relationships. That information may be built into the application by its developer, or it may be obtained from some external data source. That source may be data made available by the owner/publisher of one of the properties, or it may be made available by a third party. Services such as metadata registries can act as a source of information about properties made available by many different agencies and indeed may use that aggregated data to provide functions based on inferencing about multiple subproperty relationships.

References

[DCMIAM]

DCMI Abstract Model

<http://dublincore.org/documents/abstract-model/>

[DCMINS]

Namespace Policy for the Dublin Core Metadata Initiative (DCMI)

<http://dublincore.org/documents/dcmi-namespace/>

[DCHTML]

Expressing Dublin Core in HTML/XHTML meta and link elements

<http://dublincore.org/documents/dcq-html/>

[MARCREL]

MARC Code List: Part I: Relator Codes

<http://www.loc.gov/marc/relators/relators.html>

[RDFS]

RDF Vocabulary Description Language 1.0 (RDF Schema)

<http://www.w3.org/TR/rdf-schema/>



Metadata associated with this resource: <http://dublincore.org/documents/dc-elem-refine/index.shtml.rdf>

[Copyright](#) © 1995-2005 [DCMI](#) All Rights Reserved. DCMI [liability](#), [trademark/service mark](#), [document use](#) and [software licensing](#) rules apply. Your interactions with this site are in accordance with our [privacy](#) statements. Please feel free to [contact us](#) for any questions, comments or media inquiries.

DCMI and the DCMI Web site are hosted by [OCLC Research](#).