Title: DCMI Review of Application Profiles
Identifier: /usage/meetings/2006/04/profile-review/index.shtml

Basic documents for the review of application profiles

-- Dublin Core Application Profile Guidelines
/usage/documents/2005/09/03/profile-guidelines/

Changes needed:
-- Strengthen the requirement of assigning URIs to any non-previously-existing terms.
-- 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 in the draft Collection Description profile).

-- DCMI-compliant 'term' decision tree
http://www.ukoln.ac.uk/metadata/dcmi/term-decision-tree/

-- DC-TEXT
http://www.ukoln.ac.uk/metadata/dcmi/dc-text/

Examples

-- Simple Dublin Core
/usage/meetings/2006/04/profile-review/dcsimple/index.html

-- Collection Description AP - summary
http://www.ukoln.ac.uk/metadata/dcmi/collection-ap-summary/

-- Collection Description AP - example excerpts
/usage/meetings/2006/04/profile-review/examples/index.html

Discussion of issues

-- UB meeting discussion of profile review, Washington, May 2005
/usage/meetings/2005/09/madrid/files/2005-07-19.profile-review.txt

-- Mixing and Matching FAQ - "implementer questions about incorporating XML semantics"
http://www.ukoln.ac.uk/metadata/dcmi/mixing-matching-faq/

-- Diane on controlled vocabularies specifying obligation
/usage/meetings/2005/09/madrid/files/2005-08-06.diane-on-dcap-guidelines.txt

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

-- Dan Brickley on DCAPs
/usage/meetings/2006/04/profile-review/2005-10-05.danbri-dcap-draft.txt

Background and foundational documents (not included in packet)

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

-- DCMI Abstract Model
dublincore.org/documents/abstract-model/

-- XML, RDF, and DCAPs
www.ukoln.ac.uk/metadata/dcmi/dc-elem-prop/

-- Element Refinement in Dublin Core Metadata
dublincore.org/documents/dc-elem-refine/

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 distils the salient features of many existing profiles into a format that is as concise and simple as possible, yet as precise and detailed as is sometimes necessary to support the various uses identified above.

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

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

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

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

1 Scope

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

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

2 Definitions

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

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

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

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

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

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

3 Identifying terms with appropriate precision

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

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

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

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

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

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

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

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

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

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

4 Attributes of a Term Usage

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

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

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

4.1 Identifying attributes

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

4.2 Definitional attributes

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

4.3 Relational attributes

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

4.4 Constraints

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

5 Discussion

5.1 Descriptive Headers

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

- A brief description of the DCAP based on Dublin Core. At a minimum, the description should specify a Title, Contributor, Date, Identifier, and Description, as explained in more detail in Annex A. Ideally, the description of the DCAP will elaborate on the context in which the DCAP is intended to be used.
- Optionally, a Preamble for the DCAP should describe any technical or formatting conventions used in the DCAP. For example, if namespace prefixes are used in the Name field (see Section 5.6), these prefixes should be documented here. The Preamble can also cite Web pages or schemas in which the terms used in the DCAP are documented and defined so that such information does not need to be repeated in the "Defined By" field of each individual Term Usage; see the example in Section 6.1.1.

5.2 Readability of Term Usages

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

- Instead of creating a separate Term Usage for every Element Refinement and Encoding Scheme used in an application, such terms may simply be cited in the attributes Refined By or Has Encoding Scheme of the Term Usages of the Elements to which they refer (see Section 5.3). Note that this terser style does not support the addition of usage notes, local definitions, or annotations, for which a full Term Usage must be used.
- Attributes not needed for Term Usages can simply be omitted. At one extreme, a Term Usage might legitimately consist of just a Name and a Term URI; see the example in Section 6.1.2).
- The order of attributes presented in Section 4 is significant only for the usability of DCAPs as documents – not for future machine-processable representations of DCAPs. Authors of DCAPs may therefore change the order of attributes in the interest of readability, though they should bear in mind that any such changes may make it more difficult for people to compare two DCAPs visually. For examples of how Name (in boldface) is placed before Term URI, see Section 6.1.2 and DCMI-TERMS.
- In the interest of readability, it might make sense to describe Elements, Element Refinements, and Encoding Schemes with different subsets of relevant attributes. Indeed, these different types of terms might be grouped under separate sections of the DCAP document. (For example, see how Elements and Element Refinements are separated from Encoding Schemes in the document "DCMI Metadata Terms" [DCMI-TERMS].)
- In the interest of readability and of future machine-parsability, attributes should be repeated when necessary (as opposed to listing multiple values for a single attribute).

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

5.3.1 Using Vocabulary Terms

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

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

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

5.3.2 Using Encoding Schemes

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

- The most concise way is to use the attribute Has Encoding Scheme (or Comment) for a blanket reference to a set of encoding schemes documented elsewhere. The DCAP for Resource Discovery Network, for example, simply cites "RDN Subject Encoding Schemes" and gives a URL where the list of those encoding schemes may be found; see the example in Section 6.1.2.
- A more precise way is to use the attribute Has Encoding Scheme, repeated as necessary, to cite each Encoding Scheme by its URI (or by Name and URI); for example, see the example in Section 6.3.2.
- In addition to citing Encoding Schemes in the Has Encoding Scheme attribute of Elements, creators of DCAPs may want to describe Encoding Schemes in stand-alone Term Usages in order to annotate their usage, for example by specifying a Datatype, Occurrence, or Local Definition. The attribute Encoding Scheme For points back to the Element or Element Refinement qualified.

5.3.3 Using Element Refinements

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

- Statements such as "all terms in Vocabulary D can be used as element refinements for Contributor" can be simply recorded in a Refined By attribute (or as a Comment).
- Element Refinements can be cited one-by-one using the attribute Refined By; see the example in Section 6.3.2.
- Element Refinements can additionally be described in separate Term Usages.

5.4 Attributes copied from external sources

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

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

Where information copied from external sources is supplied, this fact should be reported in the Preamble as described in Section 5.1. Where it is necessary to distinguish in a DCAP between attributes defined locally and attributes copied from an external source, the DCAP should establish its own document-internal convention, such as distinguishing between a Local Definition and a Source Definition; see the example in Section 6.3.2.

5.5 Types of Comments

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

5.6 Term URIs versus Qualified Names

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

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

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

5.7 Declaring new elements

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

5.8 Documenting grouped or nested metadata elements

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

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

5.9 Documenting unorthodox practices

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

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

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

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

6 Examples

6.1 UK Resource Discovery Network OAI Application Profile

6.1.1 Descriptive Header

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

6.1.2 A Term Usage

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

6.1.3 Commentary

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

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

6.2 Renardus Application Profile

6.2.1 Descriptive Header

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

6.2.2 A Term Usage

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

6.2.3 Commentary

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

- The DCAP uses its own URL as an identifier.

6.3 UK e-Government Metadata Standard Application Profile

6.3.1 Descriptive Header

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

Status	Version 1.0 For publication
Subject	Metadata
Subject.category	Information management
Title	UK e-government metadata standard application profile version 1

6.3.2 A Term Usage

Term URI	http://purl.org/dc/elements/1.1/date
Defined by	http://purl.org/dc/elements/1.1/
Name	Date
Label	Date
Source Definition	A date associated with an event in the life cycle of the resource.
Source Comment	-
Local Definition	-
Local Comment: Purpose	To enable the user to find the resource by limiting the number of search hits according to a date, e.g. the date the resource was made available.
Local Comment: Notes	Dates need to appear in a format that is recognisable to people all over the world, and that can be interpreted by computer software. The W3C format allows accurate searching, and makes it clear which is the year, month or day. The format is 'ccyy-mm-dd', where 'ccyy' is the year, 'mm' is the month and 'dd' the day. When the time is also needed, add 'hh:mm', where 'hh' is the hour (using the 24 hour clock), 'mm' is minutes. More about this notation can be found at http://www.w3.org/TR/NOTE-datetime.
Local Comment: Not to be confused with	Coverage – Date refers to dates relevant to the information resource itself, not the information held within the resource. For example, for a document about the civil service in the 18th century, put '18th century' in Coverage and put the date published in Date.
Type of term	Element
Refines	-
Refined by	http://www.govtalk.gov.uk/terms/dateAcquired
Refined by	http://purl.org/dc/terms/dateAccepted
Refined by	http://purl.org/dc/terms/Available
Refined by	http://purl.org/dc/terms/dateCopyrighted
Refined by	http://purl.org/dc/terms/created
Refined by	http://purl.org/dc/terms/issued
Refined by	http://purl.org/dc/terms/dateSubmitted
Refined by	http://purl.org/dc/terms/valid
Refined by	http://purl.org/dc/terms/modified
Refined by	http://www.govtalk.gov.uk/terms/cutOffDate
Refined by	http://www.govtalk.gov.uk/terms/dateDeclared
Refined by	http://www.govtalk.gov.uk/terms/dateClosed
Refined by	http://www.govtalk.gov.uk/terms/nextVersionDue
Refined by	http://www.govtalk.gov.uk/terms/updatingFrequency
Encoding Scheme For	-
Has Encoding Scheme	http://purl.org/dc/terms/W3CDTF
Has Encoding Scheme	http://purl.org/dc/terms/Point
Similar To	-
Constraints	The value must always be taken from the specified encoding scheme, with the exception of the 'updatingFrequency' refinement.
Obligation	Mandatory
Condition	A value must be given either for the unqualified date or at least one date refinement
Datatype	-

6.3.3 Commentary

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

- Encoding Schemes and Element Refinements are listed using repeated fields in the Term Usage of the Element to which

they refer. In addition, each Encoding Scheme and Element Refinement is also described in its own Term Usage, which allows information about each of them, such as Definition and Constraints, to be recorded in the DCAP as well.
- The Term Usage marks information coming from outside sources: the "Source Definition" copies the definition of Date from DCMI documentation, while the "Local Comment" supplies usage information local to this DCAP.

Annex A: Metadata describing a DCAP

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

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

Annex B: Options for machine-interpretable DCAPs

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

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

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

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

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

Bibliography

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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>
   ```

   - If 'yes', go to question 2.
   - 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)?**
   - If 'yes', go to question 9.
   - 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>
   ```

   - If 'yes', go to question 4.
   - 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)?**
   - If 'yes', go to question 9.
   - 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>
```

- If 'yes', go to question 6.
- Otherwise, go to question 7.

6. **Are all the valid values according to the *syntax encoding scheme* simple 'value strings' (plain text strings)?**
   - If 'yes', go to question 9.
   - 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>
   ```

   - If 'yes', go to question 9.
   - 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)?**
   - If 'yes', go to question 9.
   - 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`
   - If 'yes', the thing is a valid DCMI *element*, *element refinement* or *encoding scheme*.
   - 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

Title:	**DC-TEXT: A Text Syntax for Dublin Core Metadata**
Creator:	Pete Johnston
Date Issued:	2006-03-14
Identifier:	http://www.ukoln.ac.uk/metadata/dcmi/dc-text/2006-03-14/
Replaces:	http://www.ukoln.ac.uk/metadata/dcmi/dc-text/2006-01-13/
Is Replaced By:	Not applicable
Latest Version:	http://www.ukoln.ac.uk/metadata/dcmi/dc-text/
Description of Document:	This document specifies a Text Syntax for representing Dublin Core metadata description sets.

Contents

1. Introduction

The DCMI Abstract Model [DCAM] describes the components which make up DC metadata description sets and the relationships between them. This document specifies a syntax for serialising, or representing, a DC metadata description set in plain text. The format is referred to as "DC-TEXT" A plain text format for serialisation of such description sets is useful as a means of presenting examples in a way which highlights the constructs of the Abstract Model, and also as a means of comparing the information represented in other formats such as DC-XML, RDF/XML and XHTML/HTML.

2. The DCMI Abstract Model (Summary)

According to the DCMI Abstract Model [DCAM]:

- a *description set* is made up of one or more *descriptions*
- a *description* is made up of
    - zero or one *resource URI* and
    - one or more *statements*
- a *statement* is made up of
    - exactly one *property URI* and
    - zero or one *references to* a value in the form of a *value URI*
    - zero or more *representations of* a value, each in the form of a *value representation*
    - zero or one *vocabulary encoding scheme URI*
- a *value representation* is either
    - a *value string* or
    - a *rich representation*
- a value string may be associated with a *value string language*
- a value string may be associated with a *syntax encoding scheme URI*
- a *value* may be the subject of a *related description*

3. The Syntax

A formal description of the DC-TEXT syntax is presented in Appendix A This section presents an overview of the syntax and a set of examples illustrating how the various constructs of the Abstract Model are represented.

3.1 The Structure of a DC-TEXT Document

The general structure of a DC-TEXT document is as follows:

```
namespace declaration

label (
      label ( content )
      label (
             label ( [...] )
      [ ... ]
            )
      )
```

Each of the primary components of a DC metadata description set defined by the DCMI Abstract Model is represented in DC-TEXT by a syntactic structure of the form:

```
label ( content )
```

where *label* is replaced by one of the following strings:

```
DescriptionSet, Description, DescriptionId, ResourceURI, Statement, PropertyURI, DescriptionRef,
VocabularyEncodingSchemeURI, ValueURI, ValueString, Language, SyntaxEncodingSchemeURI,
RichRepresentation, Base64, MIME
```

and *content* is either:

- a sequence of one or more syntactic structures of the form *label(content)* (i.e. these structures are "nested"); or
- a string of the form "*literal*", which represents that Unicode literal; or
- a string of the form <*uri*>, which represents a URI; or
- a string of the form *prefix*:*name*, which represents a "qualified name" used as an abbreviation for a URI
- a string which represents a language tag
- a string which is a locally-scoped identifier used to establish relationships between values and their descriptions

For each *label* value in the list above, the permitted form of *content* is determined by the syntax rules specified in Appendix A. These are explained through examples below.

The DC-TEXT syntax supports the representation of a single DC description set, so a DC-TEXT document consists of zero or more *namespace declarations* followed by a single *label(content)* syntactic structure with a *label* of DescriptionSet, and as *content*, one or more nested *label(content)* structured with a *label* of Description. i.e. a DC-TEXT document has the following outline form:

```
@prefix prefix: <uri> .

DescriptionSet (
     Description (
          Statement ( ... )
          Statement ( ... )
          )
     Description (
          Statement ( ... )
          Statement ( ... )
          )
     )
```

3.1 URIs, Qualified Names, and Namespace Declarations

The DCMI Abstract Model uses URIs to refer to resources and to metadata terms (properties, vocabulary encoding schemes and syntax encoding schemes). In the DC-TEXT syntax, URIs may be written in full or may be represented as *qualified names*. A *qualified name* is made up of two parts, a *prefix* and a *name*, separated by a colon (":"). In DC-TEXT, wherever a *qualified name* is used, it is used to represent a URI. The URI represented by the qualified name is determined by appending the name part of the qualified name to the URI with which the prefix is associated in a *namespace declaration* (sometimes called the namespace URI).

Namespace declarations occur at the start of a DC-TEXT document, and have the following form:

```
@prefix prefix: <uri>
```

For example, the following declarations associates the prefix dc with the URI `http://purl.org/dc/elements/1.1/` and the prefix ex with the URI `http://example.org/resources/`

```
@prefix dc: <http://purl.org/dc/elements/1.1/>
@prefix ex: <http://example.org/resources/>
```

Note that the limitations on the characters which can occur in the name part of a qualified name mean that there are URIs that can not be expressed as qualified names. For example the URIs `http://example.org/resources/12345` and `http://example.org/resources#12345` can not be represented as qualified names, because the name part can not include the "/" or "#" characters, and can not begin with a numeric character.

3.3 Comments

Comments can be inserted anywhere in a DC-TEXT document. A comment starts with a "#" and ends with a newline.

```
# A comment at the start of the document
@prefix prefix: <uri> .

DescriptionSet (
     Description (
# A comment at the start of a description
          Statement ( ... ) # A comment following a statement
          Statement ( ... )
          )
     Description (
          Statement ( ... )
          Statement ( ... )
          )
     )
```

3.4 String Escapes

To follow.

4. Examples

This section provides examples of how the DC-TEXT syntax represents all the constructs of the DCMI Abstract Model.

4.1 Statements Using Value Strings and Value URIs

The first example is of a description set containing a single description with a single simple statement with a property URI and a value string to represent the value:

```
DescriptionSet (
  Description (
    Statement (
      PropertyURI ( <http://purl.org/dc/elements/1.1/title> )
      ValueString ( "DCMI Home Page" )
      )
    )
  )
```

Example 1: Value Strings

The second example introduces a resource URI which identifies the subject of the description, using the `ResourceeURI ( <uri> )` syntactic structure:

```
DescriptionSet (
  Description(
    ResourceURI( <http://dublincore.org/pages/home> )
    Statement (
      PropertyURI ( <http://purl.org/dc/elements/1.1/title> )
      ValueString ( "DCMI Home Page" )
      )
    )
  )
```

Example 2: Resource URI

By introducing namespace declarations, the qualified name mechanism can be used to abbreviate both the resource URI and the property URI. The same description set as in the previous example might be encoded as follows.

```
@prefix page: <http://dublincore.org/pages/> .
@prefix dc: <http://purl.org/dc/elements/1.1/> .

DescriptionSet (
  Description (
    ResourceURI ( page:home )
    Statement (
      PropertyURI ( dc:title )
      Value String ( "DCMI Home Page" )
      )
    )
  )
```

Example 3: Qualified Names

The value string may be associated with a language tag, represented using the `Language( tag )` syntactic structure:

```
@prefix page: <http://dublincore.org/pages/> .
@prefix dc: <http://purl.org/dc/elements/1.1/> .

DescriptionSet (
  Description (
    ResourceURI ( page:home )
    Statement (
      PropertyURI ( dc:title )
      ValueString ( "DCMI Home Page"
        Language ( en-GB )
        )
      )
    )
  )
```

Example 4: Language Tags

A single statement may include multiple value strings to represent the value. In DC-TEXT this is represented by repeating the `ValueString ( "`*literal*`" )` syntactic structure:

```
@prefix page: <http://dublincore.org/pages/> .
@prefix dc: <http://purl.org/dc/elements/1.1/> .

DescriptionSet (
  Description (
    ResourceURI ( page:home )
    Statement (
      PropertyURI ( dc:title )
      ValueString ( "DCMI Home Page"
        Language ( en-GB )
        )
      ValueString ( "El Home Page de DCMI"
        Language ( es-ES )
        )
      )
    )
  )
```

Example 5: Multiple Value Strings

A statement may include a value URI to identify the value, using the `ValueURI ( <`*uri*`> )` syntactic structure:

```
@prefix page: <http://dublincore.org/pages/> .
@prefix agent: <http://example.org/agents/> .
@prefix dc: <http://purl.org/dc/elements/1.1/> .

DescriptionSet (
  Description (
    ResourceURI ( page:home )
    Statement (
      PropertyURI ( dc:title )
      ValueString ( "DCMI Home Page"
        Language ( en-GB )
        )
      ValueString ( "El Home Page de DCMI"
        Language( es-ES )
        )
      )
    Statement(
      PropertyURI ( dc:creator )
      ValueURI ( agent:DCMI )
      )
    )
  )
```

Example 6: Value URIs

4.2 Vocabulary and Syntax Encoding Scheme URIs

A statement may include a vocabulary encoding scheme URI to specify the type of the value, a class of which the value is an instance. In DC-TEXT this is represented using the `VocabularyEncodingSchemeURI ( <uri> )` syntactic structure:

```
@prefix page: <http://dublincore.org/pages/> .
@prefix agent: <http://example.org/agents/> .
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix dcterms: <http://purl.org/dc/terms/> .

DescriptionSet (
  Description (
    ResourceURI ( page:home )
    Statement (
      PropertyURI ( dc:title )
      ValueString ( "DCMI Home Page"
        Language ( en-GB )
        )
      Value String( "El Home Page de DCMI"
        Language ( es-ES )
        )
      )
    Statement (
      PropertyURI ( dc:creator )
      Value URI ( agent:DCMI )
      )
    Statement (
      PropertyURI ( dc:subject )
      VocabularyEncodingSchemeURI ( dcterms:LCSH )
      ValueString ( "Information technology")
      )
    )
  )
```

Example 7: Vocabulary Encoding Scheme URIs

A value string may be associated with a syntax encoding scheme URI, using the `SyntaxEncodingSchemeURI( <uri> )` syntactic structure:

```
@prefix page: <http://dublincore.org/pages/> .
@prefix agent: <http://example.org/agents/> .
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix dcterms: <http://purl.org/dc/terms/> .
@prefix xs: <http://www.w3.org/2001/XMLSchema#> .

DescriptionSet (
  Description (
    ResourceURI ( page:home )
    Statement (
      PropertyURI ( dc:title )
      ValueString ( "DCMI Home Page"
        Language ( en-GB )
        )
      Value String( "El Home Page de DCMI"
        Language ( es-ES )
        )
      )
    Statement(
      Property URI( dc:creator )
      Value URI ( agent:DCMI )
      )
    Statement (
      PropertyURI ( dc:subject )
      VocabularyEncodingSchemeURI ( dcterms:LCSH )
      ValueString ( "Information technology" )
      )
    Statement (
      PropertyURI ( dcterms:modified )
      ValueString ( "2006-02-14"
```

```
        SyntaxEncodingSchemeURI ( xs:date )
      )
    )
  )
```

Example 8: Syntax Encoding Scheme URIs

4.3 Multiple Descriptions in Description Set

A description set may contain multiple descriptions, represented by a list of `Description(` *content* `)` syntactic structures. The order has no significance.

```
@prefix page: <http://dublincore.org/pages/> .
@prefix dc: <http://purl.org/dc/elements/1.1/> .

DescriptionSet (
  Description (
    ResourceURI ( page:home )
    Statement (
      PropertyURI ( dc:title )
      ValueString ( "DCMI Home Page" )
      )
    )
  Description (
    ResourceURI ( page:althome )
    Statement (
      PropertyURI ( dc:title )
      ValueString ( "DCMI Alternative Home Page" )
      )
    )
  )
```

Example 9: Multiple Descriptions

A *description* may be the description of a resource which is a *value* in a statement in another description within the description set. If the resource has been assigned a URI, then that URI appears as a value URI in the statement where the resource is the value and as a resource URI in the description of that resource.

```
@prefix page: <http://dublincore.org/pages/> .
@prefix agent: <http://example.org/agents/> .
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .

DescriptionSet (
  Description (
    ResourceURI ( page:home )
    Statement (
      PropertyURI ( dc:title )
      ValueString ( "DCMI Home Page" )
      )
    Statement (
      PropertyURI ( dc:creator )
      ValueURI ( agent:DCMI )
      )
    )
  Description (
    ResourceURI ( page:althome )
    Statement (
      PropertyURI ( dc:title )
      ValueString ( "DCMI Alternative Home Page" )
      )
    Statement (
      PropertyURI ( dc:creator )
      ValueURI ( agent:DCMI )
      )
    )
  Description (
    ResourceURI ( agent:DCMI )
```

```
    Statement (
      PropertyURI ( foaf:name )
      ValueString ( "Dublin Core Metadata Initiative" )
      )
    )
  )
```

Example 10: Multiple Related Descriptions

In some cases it may be that the resources do not have URIs assigned, but such a resource may still be a value in a ststement, and the subject of another description. In DC-TEXT, the association between the value of one statement and the description of that resource is made by labelling the description using a `DescriptionId ( id )` syntactic structure. The id value may then be cited using the `DescriptionRef ( id )` syntactic structure in one or more statements elsewhere in the same description set. This is a syntactic mechnism for linking references to values to their descriptions: the id itself does not appear in the Abstrct Model

```
@prefix page: <http://dublincore.org/pages/> .
@prefix agent: <http://example.org/agents/> .
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .

DescriptionSet (
  Description (
    ResourceURI ( page:home )
    Statement (
      PropertyURI ( dc:title )
      ValueString ( "DCMI Home Page" )
      )
    Statement (
      PropertyURI ( dc:creator )
      DescriptionRef ( descDCMI )
      )
    )
  Description (
    ResourceURI ( page:althome )
    Statement (
      PropertyURI ( dc:title )
      ValueString ( "DCMI Alternative Home Page" )
      )
    Statement (
      PropertyURI ( dc:creator )
      DescriptionRef ( descDCMI )
      )
    )
  Description (
    DescriptionId ( descDCMI )
    Statement (
      PropertyURI ( foaf:name )
      ValueString ( "Dublin Core Metadata Initiative" )
      )
    )
  )
```

Example 11: Multiple Related Descriptions

4.4 Rich Representations

A value may be represented not simply by a value string, but also by a rich represntation: an XML fragment or a piece of binary data.

In DC-TEXT, an XML fragment is represented using the `RichRepresentation ( "literal" )` syntactic structure:

```
@prefix page: <http://dublincore.org/pages/> .
@prefix dc: <http://purl.org/dc/elements/1.1/> .

DescriptionSet (
  Description (
    ResourceURI ( page:home )
    Statement (
      PropertyURI ( dc:description )
      RichRepresentation ( "<p xmlns=\"http://www.w3.org/1999/xhtml\">This is the DCMI<br />Home Page</p>" )
      )
    )
  )
```

Example 12: Rich Representations - XML

In DC-TEXT, a binary data object is enocoded as a Base64-encoded literal and represented using the `Base64 ( "literal" MIME ( "MIME-type" ) )` syntactic structure:

```
@prefix page: <http://dublincore.org/pages/> .
@prefix dc: <http://purl.org/dc/elements/1.1/> .

DescriptionSet (
  Description (
    ResourceURI ( page:home )
    Statement (
      PropertyURI ( dc:description )
      RichRepresentation (
        Base64 ( "abcdefghij" MIME ( "image/png" ) )
        )
      )
    )
  )
```

Example 13: Rich Representations - Binary Data

Appendix A. Grammar

A DC-TEXT document is a sequence of Unicode characters encoded in UTF-8 defined by the grammar below. It is specified by means of the version of Extended BNF used in XML 1.0 (Third Edition) [XML]

DCAM Text - EBNF

[1]	dcTextDoc	::=	comment* ws* namespaceDeclaration* ws* comment* ws* descriptionSet ws* comment* ws*
[2]	namespaceDeclaration	::=	'@prefix' ws+ prefixName? ':' ws+ uriref
[3]	descriptionSet	::=	'DescriptionSet(' ws* comment* ws* description ws* comment* ws* ')'
[4]	description	::=	'Description(' ws* comment* ws* (descriptionId ws* comment* ws*)? (resourceURI ws* comment* ws*)? (statement ws* comment* ws*)+ ')'
[5]	statement	::=	'Statement(' ws* comment* ws* propertyURI ws* comment* ws* (vocabEncSchemeURI ws* comment* ws*)? (valueURI ws* comment* ws*)? (valueRepresentation ws* comment* ws*)* (descriptionReference ws* comment* ws*)? ')'
[6]	valueRepresentation	::=	valueString | richRepresentation
[7]	valueString	::=	'ValueString(' ws* comment* ws* quotedString ws* comment* ws* (languageTag ws* comment* ws*)? (valueURI ws* comment* ws*)? (syntaxEncSchemeURI ws* comment* ws*)? ')'
[8]	languageTag	::=	'Language(' ws* language ws* ')'
[9]	richRepresentation	::=	'RichRepresentation(' ws* comment* ws* (quotedString | Base64) ws* comment* ws* ')'

[10]	base64	::=	'Base64 (' ws* comment* ws* quotedString ws* mime comment* ws* ')'
[11]	mime	::=	'mime (' ws* quotedString ws* ')'
[12]	descriptionReference	::=	'DescriptionRef(' ws* name ws* ')'
[13]	descriptionId	::=	'DescriptionId(' ws* name ws* ')'
[14]	resourceURI	::=	'ResourceURI(' ws* resourceRef ws* ')'
[15]	propertyURI	::=	'PropertyURI(' ws* resourceRef ws* ')'
[16]	valueURI	::=	'ValueURI(' ws* resourceRef ws* ')'
[17]	vocabEncSchemeURI	::=	'VocabEncSchemeURI(' ws* resourceRef ws* ')'
[18]	syntaxEncSchemeURI	::=	'SyntaxEncSchemeURI(' ws* resourceRef ws* ')'
[19]	resourceRef	::=	uriref | qualifiedName
[20]	qualifiedName	::=	prefixName? ':' name?

The following productions are as defined by Turtle [TURTLE]:

DCAM Text - EBNF - continued

[21]	comment	::=	'#' ([^#xA#xD])*
[22]	ws	::=	#x9 | #xA | #xD | #x20
[23]	uriref	::=	'<' relativeURI '>'
[24]	language	::=	[a-z]+ ('-' [a-z0-9]+)*
[25]	nameStartChar	::=	[A-Z] | "_" | [a-z] | [#x00C0-#x00D6] | [#x00D8-#x00F6] | [#x00F8-#x02FF] | [#x0370-#x037D] | [#x037F-#x1FFF] | [#x200C-#x200D] | [#x2070-#x218F] | [#x2C00-#x2FEF] | [#x3001-#xD7FF] | [#xF900-#xFDCF] | [#xFDF0-#xFFFD] | [#x10000-#xEFFFF]
[26]	nameChar	::=	nameStartChar | '-' | [0-9] | #x00B7 | [#x0300-#x036F] | [#x203F-#x2040]
[27]	name	::=	nameStartChar nameChar*
[28]	prefixName	::=	(nameStartChar - '_') nameChar*
[29]	relativeURI	::=	ucharacter*
[30]	quotedString	::=	string | longString
[31]	string	::=	#x22 scharacter* #x22
[32]	longString	::=	#x22 #x22 #x22 lcharacter* #x22 #x22 #x22
[33]	character	::=	'\u' hex hex hex hex | '\U' hex hex hex hex hex hex hex hex | '\\' | [#x20-#x5B] | [#x5D-#x10FFFF] See String Escapes for full details.
[34]	echaracter	::=	character | '\t' | '\n' | '\r' See String Escapes for full details.
[35]	hex	::=	[#x30-#x39] | [#x41-#x46] hexadecimal digit (0-9, uppercase A-F)
[36]	ucharacter	::=	(character - #x3E) | '\>'
[37]	scharacter	::=	(echaracter - #x22) | '\"'
[38]	lcharacter	::=	echaracter | '\"' | #x9 | #xA | #xD

Notes

References

[DCAM]
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/

[TURTLE]
Turtle - Terse RDF Triple Language
http://www.dajobe.org/2004/01/turtle/

Title: Simple Dublin Core
Identifier: http://stage.dublincore.org/usage/meetings/2006/04/profile-review/dcsimple/.index.html
Source: http://stage.dublincore.org/usage/meetings/2006/04/profile-review/dcsimple/index.txt

http://dublincore.org/documents/dcmes-xml/

This document describes an encoding for the DCMES in XML subject to these restrictions:

* The Dublin Core elements described in the DCMES V1.1 reference can be used
* No other elements can be used
* No element qualifiers can be used
* The resulting RDF/XML cannot be embedded in web pages

http://dublincore.org/documents/usageguide/

The Dublin Core standard includes two levels: Simple and Qualified. Simple Dublin Core comprises fifteen elements; Qualified Dublin Core includes three additional elements (Audience, Provenance and RightsHolder), as well as a group of element refinements (also called qualifiers) that refine the semantics of the elements in ways that may be useful in resource discovery.

http://dublincore.org/documents/usageguide/glossary.shtml

Simple Dublin Core

The fifteen Dublin Core elements used without qualifiers, that is without element refinement or encoding schemes. Sometimes referred to as Dublin Core simple.

http://dublincore.org/resources/faq/

"Simple Dublin Core" is Dublin Core metadata that uses no qualifiers; only the main 15 elements of the Dublin Core Metadata Element Set are expressed as simple attribute-value pairs without any "qualifiers" (such as encoding schemes, enumerated lists of values, or other processing clues) to provide more detailed information about a resource.

http://dublincore.org/schemas/xmls/

Simple DC XML schema, version 2002-12-12

This schema defines terms for Simple Dublin Core, i.e. the 15 elements from the http://purl.org/dc/elements/1.1/ namespace, with no use of encoding schemes or element refinements.

Label	Property	Source definition	Obligation	Value String
Contributor	**dc: contributor**	An entity responsible for making contributions to the content of the resource.	optional	mandatory
Coverage	**dc:coverage**	The extent or scope of the content of the resource.	optional	mandatory
Coverage	**dc:coverage**	The extent or scope of the content of the resource.	optional	mandatory
Creator	**dc:creator**	An entity primarily responsible for making the content of the resource.	optional	mandatory
Date	**dc:date**	A date associated with an event in the life cycle of the resource.	optional	mandatory
Description	**dc: description**	An account of the content of the resource.	optional	mandatory

Format	**dc:format**	The physical or digital manifestation of the resource.	optional	mandatory
Resource Identifier	**dc:identifier**	An unambiguous reference to the resource within a given context.	optional	mandatory
Language	**dc:language**	A language of the intellectual content of the resource.	optional	mandatory
Publisher	**dc:publisher**	An entity responsible for making the resource available	optional	mandatory
Relation	**dc:relation**	A reference to a related resource.	optional	mandatory
Rights Management	**dc:rights**	Information about rights held in and over the resource.	optional	mandatory
Source	**dc:source**	A reference to a resource from which the present resource is derived.	optional	mandatory
Subject and Keywords	**dc:subject**	The topic of the content of the resource.	optional	mandatory
Title	**dc:title**	A name given to the resource.	optional	mandatory
Resource Type	**dc:type**	The nature or genre of the content of the resource.	optional	mandatory

Title: # Dublin Core Collection Description Application Profile Summary

Creator:	Dublin Core Collection Description Working Group
Date Issued:	2006-02-24
Identifier:	http://www.ukoln.ac.uk/metadata/dcmi/collection-ap-summary/2006-02-24/
Replaces:	http://www.ukoln.ac.uk/metadata/dcmi/collection-ap-summary/2005-08-25/
Is Replaced By:	Not applicable
Latest Version:	http://www.ukoln.ac.uk/metadata/dcmi/collection-ap-summary/
Status of Document:	This is a Draft DC Application Profile.
Description of Document:	This document presents a summary of the draft application profile for collection-level description developed by the Dublin Core Collection Description Working Group.

Introduction

Note: This document presents a summary view of the application profile. For full details, see the full description of the Dublin Core Collection Description Application Profile [DCCDAP].

This document describes:

1. The **set of terms** used in a class of DC metadata *description sets*.
   Specifically, it describes the set of terms used in a description of a collection (i.e. a "collection-level description", or "unitary finding aid" in the terminology of the *Analytical Model of Collections and their Catalogues* [AMCC]). A collection description conforming to this DC CD AP may be the only description in a *description set* or it may be part of a description set which includes descriptions of other resources related to the collection (as values in statements about the collection). Such other resources include the location of the collection, the services that provide access to the collection, concepts that are the subject of the collection, and other collections. While this DCAP permits the inclusion of descriptions of those related resources, it does not specify the properties and classes that may be referenced in descriptions of resources other than collections.
2. **How** the terms in this set are deployed in this class of DC metadata descriptions.
   This includes requirements for the the occurrence of *statements* using a specified *property*, constraints on the types of *value* which are referenced in a statement using a specified property (*vocabulary encoding schemes*), and constraints on the *datatypes* of the *value strings* occurring in a statement using a specified property (*syntax encoding schemes*).

The terms description set, description, property, value, vocabulary encoding scheme, value representation, rich representation, value string, syntax encoding scheme, and related description are used in the sense they are used in the DCMI Abstract Model [DCAM].

This document is **not** a description of an XML format. There may be multiple bindings of the DC CD AP to XML and to other syntaxes.

Outstanding Isssues

This DCAP is currently work-in-progress. Two major issues are still under discussion:

- Clarification (at the level of the data model) of the nature of the Location and Service entity types, and the relationships between Collection and Location and Collection and Service. The properties gen:isLocatedAt and gen:isAccessedVia currently listed below should be treated as provisional placeholders for a solution.
- How to represent the information that a collection contains some items of specified formats. The outcome of this decision may also result in changes to the representation of the information that a collection contains items of a specified type.

Vocabularies/Namespaces used in this DCAP

All references to properties and classes in DC metadata descriptions are made using URIs, In this document, Qualified Names of the form `prefix ":" local-part` are used as abbreviations for URIs which identify metadata terms. Prefixes are assumed to be associated with Namespace Names (URIs) as follows, and the corresponding URI for the term is constructed by concatenating the Namespace Name and the `local-part`:

Vocabulary Title	Namespace Name	Prefix
The Dublin Core Metadata Element Set, v1.1	http://purl.org/dc/elements/1.1/	**dc**
Dublin Core Terms	http://purl.org/dc/terms/	**dcterms**
Dublin Core Type Vocabulary	http://purl.org/dc/dcmitype/	**dcmitype**
MARC Relator Code Properties	http://www.loc.gov/loc.terms/relators/	**marcrel**
Collection Description Terms (collection-specific terms)	http://example.org/cld/terms# [temporary URI, final URI to be confirmed]	**cld**
General Description Terms (non-collection-specific terms)	http://example.org/gen/terms# [temporary URI, final URI to be confirmed]	**gen**
Collection Type Vocabulary Terms	http://example.org/cld/type# [temporary URI, final URI to be confirmed]	**cldtype**

Please note that where the Qualified Name of a term (property/class) appears against a shaded background in the table below (i.e. all terms with the prefixes `gen`, `cld` and `cldtype`), this indicates that those terms have not yet been assigned persistent URIs. Until such persistent URIs are assigned, by DCMI or by some other naming authority, these terms should be considered to be unstable and should not be referenced in metadata descriptions, except as part of the evaluation/ testing of this profile.

Property Usage

Each main row in the table below describes how a specified property should be used in a statement within a DC metadata description.

- **Label:** A short human-readable label that provides an indication of how the property is to be used in a DC CD AP collection description. The label does not appear in the description. It *may* be used to provide a descriptor for fields in displays of DC CD AP collection descriptions, but there is no requirement for display applications to use this label.
- **Qualified Name for Property:** A unique name/identifier for the property. It is presented as a Qualified Name, but is an abbreviation for a URI. This URI *must* be used to refer to the property in DC CD AP collection descriptions.
- **Usage in this DCAP:** A description of how the definition of the property is to be applied in DC CD AP collection descriptions. This information supplements the definition of the property provided by its owner/maintenance agency.
- **Obligation:** An indication of whether a statement using this property is required in a DC CD AP collection description. M = mandatory; a statement using this property is required, OR = a statement using this property is optional but recommended, O = a statement using this property is optional
- **Qualified Name(s) for Vocabulary Encoding Scheme(s):** The unique names/identifiers of classes from which values for the property should be drawn. Names are presented as Qualified Names, but are abbreviations for URIs. The URI *must* be used to refer to the class in DC CD AP collection descriptions. If no class is listed, then the DC CD AP does not specify a class from which values should be drawn. However the definition and usage of the property may determine that values of only certain types are appropriate. For example, the value of the `dc:creator` property must be an entity capable of action.
- **Value URI:** An indication of whether a value URI is to be used in a statement using the property (and vocabulary encoding scheme, where specified). M = mandatory; a value URI is required, O = a value URI is optional (see note), N = a value URI is not permitted
- **Value String:** An indication of whether a value string is to be used in a statement using the property (and vocabulary encoding scheme, where specified). M = mandatory; a value string is required, O = a value string is optional (see note), N = a value string is not permitted
- **Qualified Name(s) for Syntax Encoding Scheme(s):** The unique names/identifiers of datatypes from which value strings for the property should be drawn. Names are presented as Qualified Names, but are abbreviations for URIs. The URI *must* be used to refer to the datatype in DC CD AP collection descriptions. If no datatype is listed, then the DC CD AP does not specify a datatype from which value strings should be drawn.
- **Rich Representation:** An indication of whether a rich representation is to be used in a statement using the property (and vocabulary encoding scheme, where specified). M = mandatory; a rich representation is required, O = a rich representation is optional (see note), N = a rich representation is not permitted

Note: For each value, **at least one** of the following components must be present: a value URI, a rich representation, a value string or a (related) description.

Collection Properties

Label	Qualified Name for Property	Usage in this DCAP	Obligation	Qualified Name(s) for Vocabulary Encoding Scheme(s)	Value URI	Value String	Qualified Name(s) for Syntax Encoding Scheme(s)	Rich Represent
Collection Identifier	**dc:identifier**	A globally unique formal identifier for the collection.	OR	dcterms:URI	N	M		N
Title	**dc:title**	The name of the collection.	M		N	M		N

Alternative Title	**dcterms:alternative** (sub-property of dc:title)	Any form of name used as a substitute or alternative to the formal name of the collection	O		N	M		N
Description	**dcterms:abstract** (sub-property of dc: description)	A summary description of the content of the collection.	M		N	M		N
Size	**dcterms:extent** (sub-property of dc: format)	The size of the collection.	O		N	M		N
Language	**dc:language**	A language of the content of the items in the collection.	O	dcterms:ISO639-2	O	M		N
Collection Type	**dc:type**	A type of the collection.	O	cld:CLDType	O	M		N
Rights	**dc:rights**	A statement of any rights held in/over the collection.	O		O	O		N
Access Rights	**dcterms:accessRights** (sub-property of dc: rights)	A statement of any access restrictions placed on the collection, including allowed users, charges, etc.	O		O	O		N
Accrual Method	**dcterms:accrualMethod**	The method by which items are added to the collection.	O	cld: DCCDAccrualMethod	O	M		N
Accrual Periodicity	**dcterms: accrualPeriodicity**	The frequency with which items are added to the collection.	O	cld: DCCDAccrualPeriodicity	O	M		N
Accrual Policy	**dcterms:accrualPolicy**	The policy governing the addition of items to the collection.	O	cld: DCCDAccrualPolicy	O	M		N
Custodial History	**dcterms:provenance**	A statement of any changes in ownership and custody of the collection that are significant for its authenticity, integrity and interpretation.	O		O	M		N
Audience	**dcterms:audience**	A class of entity for whom the collection is intended or useful.	O		O	M		N
Subject	**dc:subject**	A subject or topic associated with the items in the collection.	O	dcterms:DDC	O	M		N
				dcterms:LCC	O	M		N
				dcterms:LCSH	O	M		N
				dcterms:MESH	O	M		N
				dcterms:UDC	O	M		N
Spatial Coverage	**dcterms:spatial** (sub-property of dc: coverage)	The spatial coverage of the content of the items in the collection.	O		O	M		N

Temporal Coverage	**dcterms:temporal** (sub-property of dc: coverage)	The temporal coverage of the content of the items in the collection.	O		O	M		N
Accumulation Date Range	**dcterms:created** (sub-property of dc:date)	The range of dates over which the collection was accumulated.	O	gen:RKMS-ISO8601	O	M		N
Contents Date Range	**cld: dateContentsCreated** (sub-property of dc:date)	The range of dates over which the individual items within the collection were created	O	gen:RKMS-ISO8601	O	M		N
Relationships between the Collection and Agents								
Collector	**dc:creator**	An entity who gathers (or gathered) the items in a collection together.	O		O	M		N
Owner	**marcrel:OWN**	An entity who has legal possession of the collection.	O		O	M		N
Relationships between the Collection and Location, Collection and Service								
Is Located At	**gen:isLocatedAt** (sub-property of dc: relation)	The location of the collection.	O		O	O		N
Is Accessed Via	**gen:isAccessedVia** (sub-property of dc: relation)	A service that provides access to the collection.	O		O	O		N
Relationships between Collections								
Sub-collection	**dcterms:hasPart** (sub-property of dc: relation)	A second collection contained within the current collection.	O	dcmitype:Collection	O	O		N
Super-collection	**dcterms:isPartOf** (sub-property of dc: relation)	A second collection that contains the current collection.	O	dcmitype:Collection	O	O		N
Catalogue or collection description	**cld: collectionDescription** (sub-property of dc: description)	A second collection that describes the current collection (for example, the catalogue for the current collection).	O	dcmitype:Collection	O	O		N
Associated collection	**cld:associatedCollection** (sub-property of dc: relation)	A second collection that is associated with the current collection.	O	dcmitype:Collection	O	O		N
Relationships between the Collection and other resources								
Associated publication	**dcterms: isReferencedBy** (sub-property of dc: relation)	A publication that is based on the use, study, or analysis of the collection.	O		O	O		N

References

[DCCDAP] *Dublin Core Collection Description Application Profile*
http://www.ukoln.ac.uk/metadata/dcmi/collection-application-profile/

[AMCC] Heaney, Michael. *An Analytical Model of Collections and their Catalogues*
http://www.ukoln.ac.uk/metadata/rslp/model/

[DCAM] Powell, Andy, Mikael Nilsson, Ambjörn Naeve, Pete Johnston. *DCMI Abstract Model*
http://dublincore.org/documents/abstract-model/

Changes made in this version

- Replace use of dc:relation property with cld:associatedCollection property.
- Replace use of dc:description property with cld:collectionDescription property.

Metadata associated with this resource: http://dublincore.org/documents/collection-ap-summary/index.shtml.rdf

Title:	**Dublin Core Collection Description Application Profile**
Creator:	Dublin Core Collection Description Working Group
Date Issued:	2006-02-24
Identifier:	http://www.ukoln.ac.uk/metadata/dcmi/collection-application-profile/2006-02-24/
Replaces:	http://www.ukoln.ac.uk/metadata/dcmi/collection-application-profile/2005-08-25/
Is Replaced By:	Not applicable
Latest Version:	http://www.ukoln.ac.uk/metadata/dcmi/collection-application-profile/
Status of Document:	This is a Draft DC Application Profile.
Description of Document:	This document describes the draft application profile for collection-level description developed by the Dublin Core Collection Description Working Group.

CUT

The Dublin Core Collection Description Application Profile (DC CD AP)

A DC application profile (DCAP) specifies set of terms used in a class of DC metadata *description sets*, typically the class of description sets which are deployed within a metadata application or within a set of applications and services operating within some domain or community. It describes the *properties* that are used in *statements* and how the use of those properties is constrained or adapted for the purposes of that application or domain.

So, the DC CD AP specifies;

1. The **set of terms** used in a class of DC metadata *description sets*.
   Specifically, it describes the set of terms used in a description of a collection. A collection description conforming to this DC CD AP may be the only description in a *description set* or it may be part of a description set which includes descriptions of other resources related to the collection (as values in statements about the collection). Such other resources include the location of the collection, the services that provide access to the collection, concepts that are the subject of the collection, and other collections. While this DCAP permits the inclusion of descriptions of those related resources, it does not specify the properties and classes that may be referenced in descriptions of resources other than collections.
2. **How** the terms in this set are deployed in this class of DC metadata descriptions.
   This includes requirements for the the occurrence of *statements* using a specified *property*, constraints on the types of *value* which are referenced in a statement using a specified property (*vocabulary encoding schemes*), and constraints on the *datatypes* of the *value strings* occurring in a statement using a specified property (*syntax encoding schemes*).

The terms description set, description, property, value, vocabulary encoding scheme, value representation, rich representation, value string, syntax encoding scheme, and related description are used in the sense they are used in the DCMI Abstract Model [DCAM].

The metadata terms referenced in the DC CD AP are drawn from the Dublin Core metadata vocabularies and also from other metadata vocabularies.

The DC CD AP is independent of any particular syntax for representing description sets. Separate guidelines will describe how descriptions conforming to the DC CD AP may be represented using the conventions recommended by DCMI for expressing DC metadata using the Resource Description Framework (RDF) and using the Extensible Markup Language (XML).

CUT

Vocabularies/Namespaces used in this DCAP

All references to properties and classes in DC metadata descriptions are made using URIs, In this document, Qualified Names of the form `prefix ":" local-part` are sometimes used as abbreviations for URIs which identify metadata terms. Prefixes are assumed to be associated with Namespace Names (URIs) as follows, and the corresponding URI for the term is constructed by concatenating the Namespace Name and the `local-part`:

Vocabulary Title	Namespace Name	Prefix
The Dublin Core Metadata Element Set, v1.1	http://purl.org/dc/elements/1.1/	**dc**
Dublin Core Terms	http://purl.org/dc/terms/	**dcterms**
Dublin Core Type Vocabulary	http://purl.org/dc/dcmitype/	**dcmitype**
MARC Relator Code Properties	http://www.loc.gov/loc.terms/relators/	**marcrel**
Collection Description Terms (collection-specific terms)	http://example.org/cld/terms# [temporary URI, final URI to be confirmed]	**cld**
General Description Terms (non-collection-specific terms)	http://example.org/gen/terms# [temporary URI, final URI to be confirmed]	**gen**
Collection Type Vocabulary Terms	http://example.org/cld/type# [temporary URI, final URI to be confirmed]	**cldtype**

Please note that where terms have Qualified Names with the prefixes `gen`, `cld` and `cldtype`, this indicates that those terms have not yet been assigned persistent URIs. Until such persistent URIs are assigned, by DCMI or by some other naming authority, these terms should be considered to be unstable and should not be referenced in metadata descriptions, except as part of the evaluation/testing of this profile.

Property Usage

Each table in the following section describes how a specified property should be used in a statement within a DC metadata description.

- **Identifier:** The URI by which the term is referenced in a DC metadata description.
- **Qualified Name:** The Qualified Name which is typically used as an abbreviation for the term URI.
- **Defined By:** The name and identifier of the metadata vocabulary from which the property is drawn.
- **Type of Term:** An indication of the type of the term, according to the typology of the DCMI Abstract Model
- **Source Label:** The short label provided for the property by its owner/maintenance agency.
- **Label in this DCAP:** A short label that provides an indication of how the property is to be used in a DC CD AP collection description. The label does not appear in the description. It *may* be used to provide a descriptor for fields in displays of DC CD AP collection descriptions, but there is no requirement for display applications to use this label.
- **Source Definition:** The definition provided for the property by its owner/maintenance agency.
- **Usage in this DCAP:** A description of how the definition of the property is to be applied in DC CD AP collection descriptions. This information supplements the definition of the property provided by its owner/maintenance agency.
- **Comments:** Additional information about the use of the property in the DC CD AP, typically on the values and their representation.
- **Refines:** Properties of which the current property is a subproperty.
- **Refined By:** Properties which are subproperties of the current property.
- **Uses Vocabulary Encoding Scheme:** The unique names/identifiers of classes from which values for the property should be drawn. Names are presented as Qualified Names, but are abbreviations for URIs. The URI *must* be used to refer to the class in DC CD AP collection descriptions. If no class is listed, then the DC CD AP does not specify the a class from which values should be drawn. However the definition and usage of the property may determine that values of only certain types are appropriate. For example, the value of the `dc:creator` property must be an entity capable of action.
- **Value URI:** An indication of whether a value URI is to be used in a statement using the property (and vocabulary encoding scheme, where specified). Mandatory = a value URI is required, Optional = a value URI is optional (see note), Not permitted = a value URI is not permitted
- **Value String:** An indication of whether a value string is to be used in a statement using the property (and vocabulary encoding scheme, where specified). Mandatory = a value string is required, Optional = a value string is optional (see note), Not permitted = a value string is not permitted
- **Syntax Encoding Scheme(s):** The unique names/identifiers of datatypes from which value strings for the property should be drawn. Names are presented as Qualified Names, but are abbreviations for URIs. The URI *must* be used to refer to the datatype in DC CD AP collection descriptions. If no datatype is listed, then the DC CD AP does not specify the a class from which values should be drawn.
- **Rich Representation:** An indication of whether a rich representation is to be used in a statement using the property (and vocabulary encoding scheme, where specified). Mandatory = a rich representation is required, Optional = a rich representation is optional (see note), Not permitted = a rich representation is not permitted
- **Obligation:** An indication of whether a statement using this property is required in a DC CD AP collection description. Mandatory = a statement using this property is required, Optional/Recommended = a statement using this property is optional but recommended, Optional = a statement

using this property is optional
- **Condition:** Information on any additional conditions on the obligation to use the property
- **Occurrences:** The minimum and maximum number of statements referencing this property that can occur in a metadata description

Note: For each value, **at least one** of the following components must be present: a value URI, a rich representation, a value string or a (related) description.

CUT

Subject [dc:subject]

Term Identifier	**http://purl.org/dc/elements/1.1/subject**
Qualified Name	dc:subject
Defined By	The Dublin Core Metadata Element Set, v1.1 http://purl.org/dc/elements/1.1/
Type of Term	Property (Element)
Source Label	Subject and Keywords
Label in this DCAP	Subject
Source Definition	The topic of the content of the resource.
Usage in this DCAP	A subject or topic associated with the items in the collection.
Comments for this DCAP	Keywords or subject descriptors associated with items in the collection. The terms used indicate the subject matter of the collection. Where multiple keywords or subject descriptors are provided, a separate statement should be used for each keyword or term. The vocabulary encoding schemes below are those recommended by DCMI. Other appropriate vocabulary encoding schemes may be used, and the use of a scheme must be indicated.
Refines	[n/a]
Refined by	[n/a]
Uses Vocabulary Encoding Scheme	dcterms:LCSH, Dublin Core Terms http://purl.org/dc/terms/LCSH

Value URI	Value String	Syntax Encoding Scheme(s)	Rich Represent
Optional	Mandatory		Not permitted

Uses Vocabulary Encoding Scheme	dcterms:LCC, Dublin Core Terms http://purl.org/dc/terms/LCC

Value URI	Value String	Syntax Encoding Scheme(s)	Rich Represent
Optional	Mandatory		Not permitted

Uses Vocabulary Encoding Scheme	dcterms:MESH, Dublin Core Terms http://purl.org/dc/terms/MESH **Value URI** | **Value String** | **Syntax Encoding Scheme(s)** | **Rich Represent** Optional | Mandatory | | Not permitted
Uses Vocabulary Encoding Scheme	dcterms:DDC, Dublin Core Terms http://purl.org/dc/terms/DDC **Value URI** | **Value String** | **Syntax Encoding Scheme(s)** | **Rich Represent** Optional | Mandatory | | Not permitted
Uses Vocabulary Encoding Scheme	dcterms:UDC, Dublin Core Terms http://purl.org/dc/terms/UDC **Value URI** | **Value String** | **Syntax Encoding Scheme(s)** | **Rich Represent** Optional | Mandatory | | Not permitted
Obligation	Optional
Condition	[n/a]
Occurrence	Minimum: 0, Maximum: unbounded

CUT

Encoding Schemes Used

- Uniform Resource Identifier [dcterms:URI]
- ISO 639-2 [dcterms:ISO639-2]
- Collection Type Vocabulary [cld:CLDType]
- DCCD Accrual Method [cld:DCCDAccrualMethod]
- DCCD Accrual Periodicity [cld:DCCDAccrualPeriodicity]
- DCCD Accrual Policy [cld:DCCDAccrualPolicy]
- Image [dcmitype:Image]
- Library of Congress Subject Headings [dcterms:LCSH]
- Library of Congress Classification [dcterms:LCC]
- Medical Subject Headings [dcterms:MESH]
- Dewey Decimal Classification [dcterms:DDC]
- Universal Decimal Classification [dcterms:UDC]
- Recordkeeping Metadata Schema Extension to ISO8601 [gen:RKMS-ISO8601]
- Collection [dcmitype:Collection]

Uniform Resource Identifier [dcterms:URI]

Term Identifier	**http://purl.org/dc/terms/URI**
Qualified Name	dcterms:URI
Defined By	Dublin Core Terms http://purl.org/dc/terms/
Type of Term	Vocabulary (?) Encoding Scheme
Label	Uniform Resource Identifier

Definition	Uniform Resource Identifier.
See Also	http://www.ietf.org/rfc/rfc2396.txt
Encoding Scheme for	dc:identifier, Dublin Core Metadata Element Set, v1.1 http://purl.org/dc/elements/1.1/identifier

CUT

Recordkeeping Metadata Schema Extension to ISO8601 [gen:RKMS-ISO8601]

Term Identifier	**http://example.org/gen/terms#RKMS-ISO8601**
Qualified Name	gen:RKMS-ISO8601
Defined By	General Description Terms http://example.org/gen/terms#
Type of Term	Vocabulary (?) Encoding Scheme
Label	Recordkeeping Metadata Schema Extension to ISO8601
Definition	Instances of this class are dates or periods of time represented as literals according to the rules specified in Recordkeeping Metadata Schema Extension to ISO8601.
Comments for this DCAP	[n/a]
See Also	[n/a]
Encoding Scheme for	cld:dateContentsCreated, Collection Description Terms http://example.org/cld/terms#dateContentsCreated
Encoding Scheme for	dcterms:created, Dublin Core Terms http://purl.org/dc/terms/created

---

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

    - 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).

    - Terms in non-DCMI namespaces should be documented according to DCMI requirements and have persistent URIs supported by known policies.

- All terms must have an assigned URI (by DCMI or by external organisation).
- 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 was 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)

- Overall shepherd: Andrew
- Evaluation individual terms against AM: Andrew, Andy
- Check comments: Akira
- Check documentation: Tom
- Check community buy-in: Andrew
- 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 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` (correspoinding 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" representents 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

Date: Sat, 6 Aug 2005 20:25:42 -0400 (EDT)
Subject: AP Question
From: "Diane Ileana Hillmann" <dih1@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!)

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 aginst changes in project or service URLs. The use of PURLs to identify metadata terms has already been adopted by a number of metadata-related initiatives such as DCMI itself and RDF Site Summary (RSS) 1.0 [RSS10].

Example 1: `http://purl.org/rss/1.0/link`
: RDF Site Summary is a lightweight multipurpose extensible metadata description and syndication format. The core metadata terms used by RSS are declared within an XML namespace (`http://purl.org/rss/1.0/`). For example, the property called `link` has been assigned the URI `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 identifiying 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/

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

---

Date: Wed, 5 Oct 2005 11:47:57 +0200
From: Thomas Baker <tbaker@tbaker.de>
To: Dublin Core Advisory Board <dc-ab@jiscmail.ac.uk>
Cc: Dan Brickley <danbri@W3.ORG>
Subject: Re: drafting a proposal on DC App Profiles

On Fri, Sep 16, 2005 at 09:51:40AM -0400, Dan Brickley wrote:
> I started writing something earlier today, in an attempt to flesh out
> a common understanding of what 'application profiles' might mean, and
> how they relate to the technicalities of the Abstract Model and its
> bindings, as well as to various technical components such as schema
> languages,
> query languages, and transformation technologies such as GRDDL.
>
> Posted it initially to the DC-Arch RDF taskforce list,
> http://www.jiscmail.ac.uk/cgi-bin/webadmin?A2=ind0509&L=DC-RDF-TASKFORCE&P=R2034&I=-3
>
> Here it is in initial rough form below, comments welcomed... (will move to
> public fora after the conference). I'm motivate by interest in finding out
> how I might propose an app profile that draws together DC, FOAF and other
> vocabs to address DC Agent (draft) requirements.
...
> A Proposal for Dublin Core Application Profiles
>
> Dan Brickley

All,

To put this into context, the Usage Board is focusing over the coming year on testing and refining the criteria and methods used to evaluate application profiles -- the example currently on the table is the Collection Description profile -- and the expected deliverable of that process is a revised suite of documents for supporting that process.

Foundational documents underlying the review process include the DCMI Abstract Model [1], the DCMI Namespace Policy [2], Procedure for Approval of DCMI Metadata Terms and Recommendations [3], DCMI Policy on Naming Terms [4], and some draft Guidelines for Assigning Identifiers to Metadata Terms [5], along with explanatory discussions about Element Refinement in Dublin Core Metadata [6] and about differences between XML elements and RDF properties which are relevant to the construction of application profiles [7]. These explanations are reflected in a draft Mixing and Matching FAQ [8].

As practical guides to the evaluation of application profiles, there are working drafts for Dublin Core Application Profile Guidelines [9], which specify the form and content of an application profile, and a DCMI-compliant Term Decision Tree [10], which provides a step-by-step method for checking each term in an application profile against the Abstract Model.

In Madrid, the Usage Board decided to revise [9] to reflect improvements in format devised for the presentation of the Collection Description Application Profile [11,12] and to tighten the requirement for using URIs [see Footnote 1].

In this context, Dan's draft (below) proposes some very useful additions to the types of guidance provided to developers of application profiles. The existing draft Guidelines [9] specify the form and content of a DCAP. Dan's draft suggests additional guidelines on:

-- DCAM bindings and encodings of a DCAP

    > 1. DCAM Binding Type
    >
    > A DCAM profile MAY restrict itself to a subset of the possible
    > DC Abstract Model bindings.
    ...
    > 2. Document Exemplars

```
    >
    > DCAM documentation MAY include 1 or more examplar instance documents
    > indicating typical expected usage.
    ...
    > 3. Document Instance Syntactic Restrictions (eg. DTDs)
    >
    > For DCAM bindings that specify a textual notation (typically but
    > not always XML based), any relevant syntactic profiling mechanisms
    > MAY be used.

-- Queries supported by a DCAP

    > 5. Use Case Query Patterns
    >
    > An Application Profile MAY characterise the descriptive patterns
    > important in some community of practice by expressing use cases
    > in terms of a query language (or other data access mechnism)
    > for some DCAM binding.
    >
    > 5a. Syntactic query patterns
    ...
    > 5b. Abstract query patterns

Whether as stand-alone documents or an integrated whole,
guidance on describing applications from their model and
content through bindings and encodings to supported queries
would be very useful to provide.

Since these additional issues involve the Abstract Model,
syntax, and binding, I agree that it makes sense to continue
discussion either in the RDF Task Force (where it originated)
or in DC-Architecture.

Tom

FOOTNOTES

[1] Note that this implies setting the bar somewhat higher
    than suggested by Dan in his draft:

    > 4. Namespace and Term Enumeration
    >
    > A simple technique for indicating an Application Profile is to list
    > the namespaces that are typically used in instance data.
    > The profile SHOULD indicate these namespaces using URIs.

REFERENCES

[1] http://dublincore.org/documents/abstract-model/
[2] http://dublincore.org/documents/dcmi-namespace/
[3] http://dublincore.org/usage/documents/approval/
[4] http://dublincore.org/documents/naming-policy/
[5] http://www.ukoln.ac.uk/metadata/dcmi/term-identifier-guidelines/
[6] http://dublincore.org/documents/dc-elem-refine/
[7] http://www.ukoln.ac.uk/metadata/dcmi/dc-elem-prop/
[8] http://www.ukoln.ac.uk/metadata/dcmi/mixing-matching-faq/
[9] http://stage.dublincore.org/usage/documents/2005/09/03/profile-guidelines/
[10] http://www.ukoln.ac.uk/metadata/dcmi/term-decision-tree/
[11] http://www.ukoln.ac.uk/metadata/dcmi/collection-ap-summary/2005-08-25/
[12] http://www.ukoln.ac.uk/metadata/dcmi/collection-application-profile/2005-08-25/


[[
A Proposal for Dublin Core Application Profiles

Dan Brickley


Dublin Core Application Profiles are collections of descriptive patterns,
community conventions and multi-vocabulary metadata structures
used by members of the Dublin Core community.

A variety of descriptive techniques are available to the community
in support of this, combining human oriented and mechanical
approaches depending on the level of formality, precision and machine
checkability appropriate. Application Profiles are grounded in real world
metadata practice, user needs and local priorities. As such they
```

typically express constraints and information needs that simultaneously extend and restrict global vocabulary standards such as those maintained by DC itself.

A DCAM community serves as a forum for ongoing documentation of these restrictions and extensions, relying on the DC Abstract Model (and associated formalisms such as RDF) to ensure that a consistent approach to description is taken by all Application Profiles. In the simplest case, a DC Application Profile characterises the shared description interests of some community of interest. This can be achieved using natural language and other human-oriented materials (eg. case studies, online discussion fora, etc).

In addition to human-centric documentation, Application Profiles can often usefully be described with various machine-readable techniques.

The applicability of such techniques will vary depending on the degree of consensus and commonality of interest in the relevant community. Such techniques can improve the multilingual accessibility of the profile's documentation, eg. by allowing human-oriented summaries to be automatically generated in a variety of natural languages. The remainder of this document outlines some technical approaches that can be used when documenting an application profile.

A DC Abstract Model

There are various machine-readable ways in which we can represent the descriptive patterns, community conventions and vocabulary-mixing scenarios relevant to some DC community of interest.

These techniques are usually grounded in tools and software via some specific binding of the Dublin Core Abstract Model, and MAY be grounded via a syntax-neutral abstract binding such as RDF's (ie. a concrete grounding that itself supports multiple DCAM-compatible notations).

1. DCAM Binding Type

A DCAM profile MAY restrict itself to a subset of the possible DC Abstract Model bindings.

For example, it may specify that compliant instance data be expressed using a specific named binding (such as DC-in-XHTML).

Restrictions on binding type SHOULD specify the specific version of a DCAM concrete binding.

2. Document Exemplars

DCAM documentation MAY include 1 or more examplar instance documents indicating typical expected usage.

Exemplar instances MUST be relative to named DCAM bindings, ie. if sample RDF/XML is shown, some version of a DC-in-RDF Abstract Model binding would be named.

Notes: Examplar document instances can be useful for user-facing documentation, since actual examples are often more accessible and understandable than schema-level abstractions.

3. Document Instance Syntactic Restrictions (eg. DTDs)

For DCAM bindings that specify a textual notation (typically but not always XML based), any relevant syntactic profiling mechanisms MAY be used.

For example, XML DTDs, Relax-NG or W3C XML Schemas, or Schematron schemas could all be used to express restrictions and extensions for an application profile, indicated in terms of XML-validatable rules, can be used.

Notes:
The expression of syntactic restrictions in XML, although relative to a specific textual notation, provide a powerful and industrially accepted mechanism for expression information needs, shared expectations etc.

This technique can be used against RDF/XML and non-RDF XML instance

formats.

For non-XML RDF formats, techniques such as W3C's GRDDL can be used to indicate transformation technques (eg. using XSLT) that can convert instances into DCAM-compatible abstractions and hence into other formats. These transformations can ensure that instance data can be converted into other DCAM bindings; however there is no known mechanism for translating syntactic-binding level constraints into their equivalent in other bindings.

RSS 1.0 is an example of a document format that has additional non-RDF syntactic constraints (XML nesting structures, required element patterns etc.). These conventions can be captured to varying degrees using various XML schema languages.

Note also that RDF-based profiles need not necessarily using DC terms in their instances data, since mappings to DC terms can be expressed within schemas using RDFS/OWL technology.

4. Namespace and Term Enumeration

A simple technique for indicating an Application Profile is to list the namespaces that are typically used in instance data. The profile SHOULD indicate these namespaces using URIs.

DCAPs SHOULD indicate the status of any enumerated list of namespaces, ie. whether the list is closed, open/extensible, mandatory, etc.

At a finer granularity, profiles MAY list metadata terms that are typically used in instance data. The profile SHOULD indicate these terms using URIs.

TODO: @@pointers to Schemas project etc proposals

5. Use Case Query Patterns

An Application Profile MAY characterise the descriptive patterns important in some community of practice by expressing use cases in terms of a query language (or other data access mechnism) for some DCAM binding.

5a. Syntactic query patterns

For profiles presented in terms of a concrete syntactic binding (techniques (2) and (3)), syntax-specific data matching and query languages (eg. XPath, XSLT, XQuery) MAY be used to document 1 or more common query structures. The Schematron language provides a suitable high level approach for using XPath to document metadata expectations in this way.

5b. Abstract query patterns

Profiles MAY document expected use cases in terms of abstract query patterns, ie. structured in terms of abtract metadata statements, properties and URIs rather than in terms of a textual binding of the DCAM.

Given the close relationship between DCAM and RDF, any RDF query language or data access formalism will be applicable to this task. W3C's SPARQL language offers particular facilities (such as datatyping and optional patterns) that make it a good choice here.

An Application Profile is not itself a collection of such queries. Rather, an Application Profile's documentation MAY be augmented with 1 or more indicative queries, provided as a way of linking application requirements, usage scenarios, user-needs with formally represented patterns of properties and metadata statements.

For example, the following query indicates a pattern for using Dublin Core, FOAF and SKOS vocabularies together for the purpose of finding the interests of colleagues of some specified Agent. Note that this is a highly task-centric pattern, and not an general abstract pattern. Profile documentation MAY present a range of indicative queries, illustrating both specific scenarios and more general vocabulary combination patterns.

(@@todo: fix SKOS stuff to be accurate!)

eg. 1: AP Use Case - "combining DC, FOAF + SKOS for
locating colleague subject interests"

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX skos: <http://www.w3.org/2002/skos-namespace-@@@/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

SELECT ?ilabel ?iblurb ?wphp ?name

WHERE
{
   [ rdf:type foaf:Agent;
     foaf:name ?name;
     foaf:workplaceHomepage ?wphp;
     foaf:homepage [ dc:subject [ skos:label ?ilabel; skos:scopeNote ] ]
   ]
}
```

This says, in effect,

"Find us values for ?ilabel, ?iblurb, ?wphp, ?name where
there is something with a name we call ?name
that is of type Agent, that has a workplace homepage
that is ?wphp, and find us SKOS labels and scopenotes for the dc:subject
of that page".

Note that this:

- combines multiple vocabulary namespaces
- describes exact patterns for some (but not all!) ways in which such namespaces can be combined
- binds to the DC Abstract Model via RDF
- and hence binds to any concrete DCAM syntax
- corresponds directly to machine-usable queries, for any data that can be transformed into RDF metadata statements (eg. via GRDDL per (3) above, for non-RDF/XML notations).

The results of such a query are sets of variable-value associations,
for example some results of this query above might be:

```
ilabel: Carpentry
iblurb: Making things with wood
name: Eric Miller
wphp: http://www.w3.org/

ilabel: Photography
iblurb: Taking blurry photos
name: Dan Brickley
wphp: http://www.w3.org/
```

...assuming a dataset in which SKOS, DC and FOAF were combined
appropriately, ie. FOAF for people stuff, DC for 'subject' (we could
also have asked for titles, descriptions etc of the agent's homepage),
and SKOS to elaborate on the details of the subject of the homepage).
(the simplistic scenario assumption here is that people are
interested in the topics their homepages cover...).

]]

--
Dr. Thomas Baker baker@sub.uni-goettingen.de
SUB - Goettingen State +49-551-39-3883
and University Library +49-30-8109-9027
Papendiek 14, 37073 Göttingen