

ALA American Library Association

# METADATA APPLICATION PROFILES

Theodore Gerontakos and  
Benjamin Riesenberg

IN COLLABORATION WITH CORE PUBLISHING

**Library Technology Reports**

Expert Guides to Library Systems and Services

AUG/SEPT 2021  
Vol. 57 / No. 6  
ISSN 0024-2586

# Library Technology

R E P O R T S

Expert Guides to Library Systems and Services

## **Metadata Application Profiles**

*Theodore Gerontakos and Benjamin Riesenberg*

IN COLLABORATION WITH CORE PUBLISHING



ALA TechSource

American Library Association

# Library Technology REPORTS

ALA TechSource purchases fund advocacy, awareness, and accreditation programs for library professionals worldwide.

## Volume 57, Number 6

Metadata Application Profiles  
ISBN: 978-0-8389-4826-2  
DOI: <https://doi.org/10.5860/ltr.57n6>

## American Library Association

225 N. Michigan Ave., Suite 1300  
Chicago, IL 60601-7616 USA  
[alatechsource.org](http://alatechsource.org)  
800-545-2433, ext. 4299  
312-944-6780  
312-280-5275 (fax)

## Advertising Representative

Patrick Hogan  
[phogan@ala.org](mailto:phogan@ala.org)  
312-280-3240

## Editor

Patrick Hogan  
[phogan@ala.org](mailto:phogan@ala.org)  
312-280-3240

## Copy Editor

Judith Lauber

## Production

ALA Production Services

## Cover Design

Alejandra Diaz and ALA Production Services

*Library Technology Reports* (ISSN 0024-2586) is published eight times a year (January, March, April, June, July, September, October, and December) by American Library Association, 225 N. Michigan Ave., Suite 1300, Chicago, IL 60601-7616. It is managed by ALA TechSource, a unit of the publishing department of ALA. Periodical postage paid at Chicago, Illinois, and at additional mailing offices. POSTMASTER: Send address changes to *Library Technology Reports*, 225 N. Michigan Ave., Suite 1300, Chicago, IL 60601-7616.

Trademarked names appear in the text of this journal. Rather than identify or insert a trademark symbol at the appearance of each name, the authors and the American Library Association state that the names are used for editorial purposes exclusively, to the ultimate benefit of the owners of the trademarks. There is absolutely no intention of infringement on the rights of the trademark owners.



ALATechSource

Copyright © 2021  
Theodore Gerontakos and Benjamin Riesenberg  
All Rights Reserved.

## About the Authors

**Theodore Gerontakos** is the head of the Metadata and Cataloging Initiatives Unit at the University of Washington Libraries, where he also served as a metadata librarian from 2004 to 2019. He has participated in and advised hundreds of projects, most including some data modeling or application profile development. His current interest is in escorting library data into the Semantic Web using multiple standardized data models and other standards and best practices.

**Benjamin Riesenberg** has served as metadata librarian at the University of Washington Libraries since 2019. They became interested in application profiles for publishing digital-collections metadata as linked open data while completing the MLIS program at the University of Washington iSchool. They are currently working to develop RDA/RDF metadata application profiles for evaluation in cataloging workflows and implement profiles for use with digital collections.

## Abstract

In *Library Technology Reports* (vol. 57, no. 6), “Metadata Application Profiles,” metadata application profiles (MAPs) are discussed in two broad categories depending on whether or not they adhere to linked-data practices. There exist a broad range of purposes for MAPs, including metadata implementation and interoperability. MAPs have four components: the application, entities, properties, and values. MAP creators gather MAP components from already existing sources, including ontologies, schemas, vocabulary encoding schemes, and syntax encoding schemes. Implementers may present MAPs in human-readable, machine-readable, and hybrid formats. Several examples in the text demonstrate specific MAP features.

## Subscriptions

[alatechsource.org/subscribe](http://alatechsource.org/subscribe)

# Contents

<b>Chapter 1—Introduction</b>	<b>5</b>
<i>Benjamin Riesenber</i>	
<b>Two Broad Categories</b>	5
<b>Metadata</b>	5
<b>Note</b>	7
<b>Chapter 2—Purposes</b>	<b>8</b>
<i>Benjamin Riesenber</i>	
<b>Metadata Implementation</b>	8
<b>Interoperability</b>	10
<b>Supporting External Applications</b>	12
<b>Notes</b>	13
<b>Chapter 3—Components</b>	<b>14</b>
<i>Theodore Gerontakos</i>	
<b>The Application</b>	14
<b>Entities</b>	15
<b>Properties</b>	18
<b>Values</b>	19
<b>Notes</b>	21
<b>Chapter 4—Sources</b>	<b>23</b>
<i>Theodore Gerontakos</i>	
<b>Ontologies</b>	23
<b>Schemas</b>	24
<b>VES</b>	25
<b>SES</b>	25
<b>Finding Sources</b>	26
<b>Notes</b>	26
<b>Chapter 5—Presentations</b>	<b>28</b>
<i>Benjamin Riesenber</i>	
<b>Natural Language MAPs</b>	28
<b>Encoded MAPs</b>	29
<b>Hybrid MAPs</b>	30
<b>Notes</b>	32

## Contents, continued

<b>Chapter 6—Examples</b>	<b>33</b>
<i>Theodore Gerontakos</i>	
<b>Fragment 1: Professional-Level Machine-Readable AP</b>	<b>34</b>
<b>Fragment 2: Professional-Level Human-Readable AP</b>	<b>36</b>
<b>Fragment 3: Widely Adopted Human-Readable AP for Library Data</b>	<b>37</b>
<b>Fragment 4: Platform-Specific AP</b>	<b>38</b>
<b>Fragment 5: Domain Model</b>	<b>38</b>

# Introduction

Benjamin Riesenberg

The purpose of this issue of *Library Technology Reports* is to provide a brief introduction to metadata application profiles (MAPs) and to present information about what they do, how they are put together, and how they are changing. In providing examples of MAPs currently in use and development, the issue also shows how they fit into the larger context of library work, specifically within technical services work taking place in cataloging and metadata units at many institutions.

## Two Broad Categories

One challenge in discussing metadata application profiles is taking into account the diversity of practices that they can be used to define. When you consider that each collection of information resources is different, that resources and their descriptions are managed and accessed using myriad software applications, and that information is stored in a growing and evolving array of data formats, it is easy to see that permutations of these factors add up to an overwhelming variety of scenarios.

In much of the discussion here, this wide variety of metadata implementations has been simplified into two broad groups: linked-data implementations and everything else. Both categories can involve different descriptive practices, data formats, and software applications, but considering practices as they fall into these two broad categories is helpful in discussing many aspects of metadata and metadata modeling, and it is often necessary to differentiate between them when discussing MAPs. For example, the ways that entities—the resources that are described by metadata—are handled (or not) in linked-data applications versus non-linked-data applications can be quite different.

For readers who want to acquire basic knowledge of the Resource Description Framework (RDF), the following texts offer helpful starting points. The “RDF 1.1 Primer” in particular is an accessible introduction to the basic concepts and provides helpful examples.

Schreiber, Guus, and Yves Raimond, eds. “RDF 1.1 Primer.” W3C Working Group Note, World Wide Web Consortium. June 24, 2014. <https://www.w3.org/TR/rdf11-primer>.

Cyganiak, Richard, David Wood, and Markus Lanthaler, eds. “RDF 1.1 Concepts and Abstract Syntax.” W3C Recommendation, World Wide Web Consortium. February 25, 2014. <https://www.w3.org/TR/rdf11-concepts>.

A certain level of familiarity with essential linked-data concepts is assumed in these chapters, but in-depth knowledge of data models, implementations, syntaxes, and applications is not required. The Resource Description Framework (RDF) is a primary—but not the only—data model underlying linked-data and Semantic Web practices, and this term is used often throughout the issue.

## Metadata

A MAP may go by different names—variations such as *application profile* or simply *profile*, or alternate terms such as *data dictionary* or even *data model*. A MAP can provide a model—that is, a detailed definition of the standards, tools, and practices used to provide description and access for information resources. A MAP is not only detailed, but also complete, incorporating the three primary things that most metadata

```

<metadata>
  <record>
    <displayedAt>https://upload.wikimedia.org/wikipedia/commons/3/39
    /PDX_palms.jpg</displayedAt>
    <title>Northwest U.S. Palm Trees</title>
    <creator>Riesenberg, Benjamin</creator>
    <location>Portland, Oregon</location>
    <date>2016-04-17</date>
    <type>StillImage</type>
  </record>
</metadata>

```

**Figure 1.1**  
An XML metadata instance, showing metadata elements and values

```

@prefix ex: <http://example.org/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix schema: <https://schema.org/> .
@prefix dct: <http://purl.org/dc/terms/> .
@prefix dce: <http://purl.org/dc/elements/1.1/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

ex:a_photo rdf:type schema:MediaObject .
ex:a_photo schema:contentUrl
<https://upload.wikimedia.org/wikipedia/commons/3/39/PDX_palms.jpg> .
ex:a_photo dct:title "Northwest U.S. Palm Trees"@en .
ex:a_photo dce:creator "Riesenberg, Benjamin"@en .
ex:a_photo dct:spatial <http://vocab.getty.edu/tgn/7014273> .
ex:a_photo dct:date "2016-04-17"^^xsd:date .
ex:a_photo dct:type <http://purl.org/dc/dcmitype/StillImage> .

```

**Figure 1.2**  
An RDF metadata instance, showing triples with subjects, predicates, and objects

instances—that is, a given quantity of metadata—include: the entities described, the properties used to describe them, and the values provided for these properties. While each of these will be covered in detail in subsequent chapters, it would be difficult to begin describing the MAP-making process without first saying a bit about each.

## Entities

Metadata describes entities. Records in a library’s online public-access catalog describe printed books and other physical media on the shelves, e-books available through a web browser, and many more types of resources; metadata in a digital-collections platform describes images, sound recordings, documents, and more; attributes and values in an online shopping website describe products for sale.

In non-RDF metadata implementations, the entities themselves may be somewhat difficult to find in a metadata instance. In some digital-collections

platforms, for example, the entity—for example, a digital image—doesn’t really exist in the metadata, only in the user interface. We may export metadata from such a system and not really see the entity, only the statements made about it. An XML metadata export, for example, may look something like figure 1.1.

In RDF implementations, the entities being described are somewhat easier to find. Because RDF uses a “triple” structure to make statements, with each triple including a subject, predicate, and object, we can clearly identify the entity being described in each triple. This may be easier or harder based on the serialization or syntax used to encode the data, but in the example in figure 1.2, which uses Turtle syntax, we can see a subject—an entity—quite clearly in each statement.<sup>1</sup>

## Properties

Properties, which may also be referred to using terms such as *attribute* and *field*, are descriptive elements

(elements is another term used here and elsewhere to refer to them) used to make statements about entities. The nature of properties, like that of entities, also differs in linked-data and non-linked-data implementations. Most user interfaces presenting metadata will provide textual labels for properties so that users can understand the resource descriptions. But exports or other views of a metadata instance underlying these displays will vary widely.

In an RDF metadata instance, each property must be identified by either a Uniform Resource Identifier (URI) or an Internationalized Resource Identifier (IRI; the term we will most often use here). Best practices for creating and publishing RDF vocabularies require that these IRIs can be looked up (dereferenced) via the World Wide Web to retrieve information about the resources they identify. User interfaces presenting RDF data to humans will most likely display textual labels for properties, but in the underlying data these properties will be identified by machine-actionable IRIs. In figure 1.2 we see an entity, property, and value in each triple. Entities, properties, and some values are IRIs—often prefixed names, such as `rdf:type`, used in place of full IRIs—while other values are textual or numeric.

## Values

The properties we use to describe entities are somewhat meaningless unless we provide values for them. MAPs can define values in a metadata instance by putting constraints in place. Examples of these constraints are requirements that all books have one and only one value for the title property, that values for a creator property be recorded using a “[last name], [first name]” format, or that creator values be taken from the Library of Congress Name Authority File wherever available.

Depending on the nature of an implementation, these constraints may include human-readable instructions for creating or transcribing values, specifications for formats to which values must adhere, or specific sources from which values must be taken.

## Note

1. Eric Prud’hommeaux and Gavin Carothers, eds., “RDF 1.1 Turtle: Terse RDF Triple Language,” W3C Recommendation, World Wide Web Consortium, February 25, 2014, <https://www.w3.org/TR/turtle>.



# Purposes

Benjamin Riesenber

**M**APs serve a range of purposes and offer benefits for metadata creators, managers, and consumers. They can provide a detailed model for reference, a resource that exists outside of the collections, institutions, and technical platforms within which metadata exists. This resource aids the metadata design and creation processes and enables the accurate interpretation of metadata following creation, allowing reuse in new environments. MAPs may be repurposed and reused in whole or in part, saving time and effort and improving metadata quality and consistency across collections, institutions, and platforms. These benefits are discussed in more detail below, along with recommendations for designing and implementing MAPs that best realize them.

## Metadata Implementation

### Benefits during Metadata Design

MAP implementation provides benefits from the beginning of the metadata life cycle. The components of a MAP can provide a clear checklist for completion during the metadata design process. Through the process of defining entities, properties, and values, the many decisions required in order to create a complete metadata model are partitioned into more manageable sets. We can work through a logical series of questions built around the structure of the MAP itself. For example:

- What kind of entities will be described? Do these need to be specifically defined for our application? If so, where are our resource types defined in existing vocabularies, and which definitions

### Vocabularies

Entities and properties for use in a MAP may be drawn from a diverse array of sources. These are often referred to here simply as vocabularies, but they may also be called schemas, ontologies, taxonomies, or thesauruses. These sources themselves can be modeled, described, and documented in many ways. They will be considered in more detail in the chapters on MAP components and sources.

best meet the needs of our application?

- What kind of information do we need to provide about our resources—for discovery, administration, reuse, or other purposes? What properties will we need in order to do this? Can existing elements meet all of our needs?
- How should the values for each property be structured? What values should be required for all resources? What are the local needs for searching and display of these values? When specifying constraints on values, will we prioritize local needs, ease of data reuse, or other criteria?

Because MAPs can define or constrain entities, properties, and values, working through a MAP design process can help ensure that metadata planning is comprehensive. A draft MAP can be shared with stakeholders for feedback prior to metadata creation to better ensure that resource description will meet the needs of users.

### MAP REUSE

Publishing a MAP by making it openly accessible online or sharing it with institutional stakeholders

	Date
<b>Instructions</b>	Enter dates using the following format: YYYY-MM-DD
<b>Examples</b>	For March 12, 1976, enter: 1976-03-12
	For December 1, 2020, enter: 2020-12-01

**Figure 2.1**  
Data-entry instructions and examples in a human-readable MAP

and engaging with comments and feedback helps move our work toward shared best practices and build consensus with regard to metadata modeling. The fact that MAPs are reusable, either in part or as a whole, is also beneficial. Use or adaptation of an existing MAP for new work can help institutions improve descriptive consistency. This reuse also further propagates best practices and can save time and effort during the metadata design process.

### Benefits during Metadata Creation

If they provide a detailed model and clear guide for generating resource descriptions, MAP implementations will yield significant benefits during metadata creation, whether they are designed for use by humans or for machine processing. Some of these benefits are described below, and a more detailed look at human- and machine-readable MAP presentations is provided in chapter 5.

#### HUMAN-READABLE MAPS

Human-readable MAPs are a way to present guidance to metadata creators such as catalogers and other specialists as they work, providing clear and precise rules, answering questions, and resulting in more consistently formed metadata values.

For example, in a human-readable MAP, the specification that values for a date property be formatted in a certain way can be supplemented with example values (see figure 2.1).

By providing clear guidance and examples, human-readable MAPs improve workflows for metadata creation by anticipating and answering questions that come up during the process. If such MAPs provide clear guidance, if they are fully integrated into metadata-creation workflows and used by the staff doing the work, they improve the quality and consistency of metadata values, assisting users by improving the functionality of metadata in discovery systems.

The implementation of a human-readable MAP also provides an opportunity to receive feedback from staff creating resource descriptions. Any differences between what the MAP requires and what can be

generated using local tools for metadata creation will be readily apparent to staff doing the work.

#### MACHINE-READABLE MAPS

Human-readable MAPs provide information to catalogers and metadata specialists who are creating metadata, so the conformance of values with constraints depends on the extent to which guidance is understood and adhered to during descriptive work. In contrast, a machine-readable MAP may interact directly with an application used for data entry. Instead of or in addition to providing instructions for the entry of a date value, for example, this approach may generate a data-entry form that does not accept a date value unless it is entered in the YYYY-MM-DD format.

The Sinopia Linked Data Editor offers examples of the ways in which a machine-readable MAP can pass information to a data-entry form.<sup>1</sup> To create resource descriptions with the Editor, a user must first select a MAP, referred to as a resource template within the Sinopia platform. These resource templates encode MAP information in machine-readable form, identifying a single entity type that the template may be used to describe and enumerating properties that may be used to describe it. The resource template also defines constraints on values for each property, such as:

- whether a value is required
- whether multiple values can be entered
- whether the data-entry form should be set up for inputting a literal value (textual or numeric string) or an IRI
- which vocabulary encoding schemes (controlled vocabularies) should be queried for users to select an IRI from

Because linked-data bibliographic descriptions constructed using data models such as BIBFRAME and RDA/RDF require description of multiple conceptual entities to catalog a single item in hand, such as a print monograph, multiple resource templates are

### Vocabulary Encoding Schemes

Some readers may be unfamiliar with the term *vocabulary encoding scheme* and instead accustomed to using the term *controlled vocabulary* to refer to a controlled list of values that may be selected from for use in metadata instances. The Library of Congress Name Authority File is a well-known example. In non-RDF implementations, it is common to specify that textual values should be entered as they are found in a vocabulary encoding scheme (VES). In RDF metadata instances, an IRI will often be retrieved from a VES for use as a value.

## Syntax Encoding Schemes

A syntax encoding scheme (SES) provides specific rules for formatting textual or numeric values (called literals in RDF implementations). The XML Schema datatypes, which include technical specifications for formatting a variety of values such as numbers and dates, are well-known examples.

used to describe a single resource.

Machine-readable MAPs also provide a path back to the sources of MAP components. A Sinopia resource template, for example, includes IRIs for all entities and properties, and each of these can be dereferenced to access further information about source vocabularies.

## Interoperability

Because MAPs can provide a complete and detailed specification of a metadata model, they are essential in supporting interoperability. The Dublin Core Metadata Initiative glossary definition for *interoperability* is “the ability of two or more systems or components to exchange descriptive data about things, and to interpret the descriptive data that has been exchanged in a way that is consistent with the interpretation of the creator of the data.”<sup>2</sup> Following this two-part definition, we can consider interoperable metadata as that which meets two distinct requirements:

- *Semantic interoperability*: The meanings of entities, properties, and values may be accurately understood by someone wishing to reuse metadata in an external application.
- *Technical interoperability*: Encoded metadata is compatible with an external application for reuse, or the original encoding can be successfully interpreted and processed for reuse.

In order for a MAP to play its role in supporting interoperability, effort is required during the design phase, prior to the point of metadata creation. This is needed to answer questions related to each component of a MAP and define each. MAPs resulting from this work serve as guides for local metadata creators and managers who will generate and publish metadata instances, as well as for external stakeholders who will reuse them.

MAP design can best support interoperability by addressing all three components of a metadata model: entities, properties, and values. The definition of entities and properties alone is not sufficient for the goal of supporting metadata reuse in external systems. To support accurate interpretation and reuse of a

metadata instance, a MAP must include well-defined value constraints. This may include requiring that textual or numeric values conform to syntax encoding schemes (SESs) such as data types or other formatting rules, that they come from certain vocabulary encoding schemes (VESs), that values for a given property are required or may or may not be repeated, and so on.

## Assessing Source Vocabularies to Support Interoperability

To model all components of a MAP, implementers will need to assess published vocabularies and select needed elements. By viewing vocabularies and applying criteria from the perspective of interoperability, it is possible to build a MAP using components that support the accurate interpretation and reuse of metadata. Three such criteria are given below and can be applied when assessing sources for entities, properties, syntax encoding schemes, and vocabulary encoding schemes.

*IS THE SOURCE WELL-KNOWN AND WIDELY USED IN THE RELEVANT KNOWLEDGE DOMAIN?*

Do many collections in the same domain use terms from the vocabulary? The degree to which source vocabularies are well-known and well-used in relevant knowledge domains is an important factor in selection. Metadata modeled using well-known standards is more likely to be reusable in a wide variety of external applications.

For some implementations general-purpose terms may be desired. The use of properties such as the Dublin Core Metadata Initiative (DCMI) Elements or Terms, which are less specific to any particular knowledge domain and more broadly applicable, may be preferable for some collections.

*IS THE SOURCE CLEARLY DEFINED? IS A PUBLISHED DEFINITION READILY AVAILABLE ONLINE?*

Clear definitions of entities, properties, SESs, and VESs provide a basis for agreement on usage among implementers, and this consistent usage enables interoperability.

Terms should be described in clear language. For an SES specifying data types, technical definitions should be provided. Linked-data terms should be modeled in RDF using OWL, RDF Schema, SKOS, or another data-modeling vocabulary or ontology language. It is highly preferable that these definitions be published in well-maintained and easily findable websites that include information about the organization that publishes them (see figure 2.2).

RDA Registry Elements Values Data Tools Releases About FAQ Guide Project RDA Toolkit

## RDA element sets

### Work properties

Properties that represent the attribute and relationship elements of the RDA Work entity.

Each property in the element set:

- has a domain of the class that represents the Work entity.
- is linked from its child **datatype** property in [RDA Work datatype properties](#) by *rdfs:subPropertyOf*.
- is linked from its child **object** property in [RDA Work object properties](#) by *rdfs:subPropertyOf*.

Number of active elements: 590

Namespace:	<a href="http://rdaregistry.info/Elements/w/">http://rdaregistry.info/Elements/w/</a>
Suggested prefix*:	rdaw
Example curie:	rdaw:P10001

\* registered at [prefix.cc](#)

### Downloads

- CSV (text/csv) (English language only)
- JSON-LD (application/json | application/json+ld)
- N-Triples (text/rdf+nt)
- RDF/XML (application/rdf+xml)

### Languages

Catalan Danish Dutch **English** Finnish French German Hungarian Norwegian Swedish Vietnamese

### Properties Index

Show  entries

Search:



#	CURIE	Label	Definition	Subproperty of
#	<a href="#">rdaw:P10061</a>	"has author agent"	"Relates a work to an agent who is responsible for creating a textual work."	<a href="http://rdaregistry.info/Elements/w/P10065">http://rdaregistry.info/Elements/w/P10065</a> ["has creator agent of work" (en)]

Showing 1 to 1 of 1 entries (filtered from 607 total entries)

Previous  Next

RDA Vocabularies and RDA Registry are licensed under a [Creative Commons Attribution 4.0 International License](#). Based on a work at <http://rdaregistry.info>.

This page was last updated 26 Jan 2021.

**Figure 2.2** A property definition viewed online at the RDA Registry, providing a brief overview and links to download RDF definitions

### IS THE SOURCE STABLE?

If the vocabulary has a history of development, this may be a positive indicator of high visibility and engagement with a community of use. However, changes made to term definitions have the potential to change the interpretation of metadata values created using them. If development is continuing—for example, if multiple versions of a vocabulary have been released—is information about these changes available?

### Creating New Terms

In some cases, implementers may not be able to find needed components in published vocabularies. For a project creating descriptions of resources in specialized knowledge domains or one that must meet unique local needs, implementers may find that needed entity types, properties, VESs, or other terms have not

been defined in existing sources. In such cases, they may wish to create new properties to record unique or highly specific attributes, or define a new RDF resource class for description, or create other new terms for use.

To support data reuse in such a case, implementers can provide and publish term definitions in a way that meets the same requirements as those for existing vocabularies. In the case of new RDF classes, properties, or other resources, this means modeling using a linked-data modeling language such as RDF Schema, SKOS, or OWL. Importantly, these modeling tools allow for describing relationships to existing terms, for example, as subclasses and subproperties of existing classes and properties. The identifiers for new terms should be dereferenceable on the web, allowing users of the metadata to access the definitions for each. In non-RDF implementations, clear and easily accessible definitions for new properties, as well as for

any local controlled vocabularies or value-formatting rules, will be important for users to make sense of the data and reuse it in the future.

### **Additional Considerations for Assembling MAP Components**

Approaching the selection of MAP components using the criteria outlined above supports the creation of well-defined metadata that can be accurately interpreted and successfully reused. Moving into completing the assembly and implementation of a MAP, there are additional considerations. These are based upon the selection of MAP components and concern the ways in which they fit together.

#### *PAIRING ENTITIES AND PROPERTIES*

Working through entities, properties, and values—the three facets of a metadata model that a MAP can define—and how they fit together, it may be best to start with the relationship of entities to properties. When combining published entity classes (in the case of RDF implementations) and properties, the question of whether or not a given property may be used with a given entity requires looking at the specifications provided in the source for each.

In non-RDF implementations, the usage of properties to describe a given resource type should not conflict with published definitions or guidance for the properties in question, or with a commonsense understanding of the entity and properties alike. In an RDF implementation, property definitions will often include a domain, indicating a resource class or classes that a property may be used to describe. Note that these may also have subclasses that will fall within the property's domain. This value should be checked against the resource type of the entity that the property is to be used to describe.

#### *PAIRING PROPERTIES AND VALUE CONSTRAINTS*

It is essential that definitions and guidance for properties do not conflict with those for the SESs and VESs used to constrain their values. In non-RDF implementations, the combination of a given property with a value constraint or constraints should not conflict with a commonsense understanding of the meaning of the property and its expected value. In linked-data implementations, property definitions will often include a range, indicating the class or classes of RDF resources that are allowed as a value for the property. If IRIs from a given VES will be used with the property, the class or classes defined for those resources should be checked against the property's range.

From the perspective of data reuse, any constraints included in published definitions of properties

can be considered a minimum or baseline to which values are expected to conform. Implementing value constraints for a property in a MAP that match or are more restrictive than those in the property's definition can be expected to facilitate interoperability for a created metadata instance. Requirements or constraints that contradict those provided in published definitions can be expected to cause problems for data reuse.

### **Supporting External Applications**

Much of the work required to support the creation of metadata instances that can be accurately interpreted and reused successfully takes place as a MAP is designed for use, prior to the creation of any metadata. The benefits of this work are realized after the creation of metadata instances based on the MAP, when metadata is harvested or exported and processed for reuse in an external system. Metadata instances that conform to MAPs designed as described above will be clearly defined semantically and technically and well-suited for reuse.

### **Supporting Semantic Interoperability**

If the meaning of values in a metadata instance can be accurately interpreted by agents who wish to reuse it in new applications or environments, it is semantically interoperable. This interpretation should be possible given a reasonable amount of effort and will be based not only on the instance itself but also on documentation such as a MAP. The following characteristics of metadata instances created in conformance with MAPs support semantic interoperability:

- Clear identification of MAP components adopted for use from external vocabularies, enabling agents wishing to reuse metadata to access source definitions.
- The availability of definitions for MAP components—entities, properties, SESs, and VESs—in these published sources.
- When a MAP specifies that values must come from a particular VES, the degree to which IRIs are entered accurately and match the source, or to which textual representations match the source exactly, can allow for future disambiguation of literal values and reconciliation with source vocabularies.
- The absence of conflicts with published definitions for entities, properties, and value constraint components that have been combined in a MAP.
- The extent to which catalogers and specialists creating metadata understood and adhered to MAP component definitions and value constraints, or

the extent to which a machine-readable MAP passed specifications to a data entry form or validation tool to restrict nonconformant values.

It is useful to consider technical or syntactic interoperability separately from semantic interoperability for several reasons. The applications and data formats used to manage and serve metadata are distinct from semantic models. Additionally, these applications and formats evolve rapidly and independently of semantic models and vocabularies.

### Supporting Technical Interoperability

Technical interoperability may be defined as the availability of a metadata instance in an encoded format that can be ingested, displayed, queried, or otherwise used by an external application. As differing applications require differing data formats, there will be many cases in which data for reuse will not be compatible with a new application. In these cases, clearly defined technical constraints in a MAP, such as SESs, will aid the processing of a metadata instance for reuse.

Semantic interoperability requires the meaning of and relationships between entities, properties, and values in a metadata instance to align with published definitions for the entities, properties, VESs, and SESs that were used to create it. Technical interoperability requires entities, properties, and values in an instance

to be encoded and formatted in a way that can be used by external systems or interpreted accurately to allow for reuse with a reasonable amount of processing. As a starting point, these must be available in a character encoding standard such as UTF-8 that can be read by the new system. In RDF implementations, it will be essential that all IRIs for entities, properties, and values be valid and actionable as Uniform Resource Locators (URLs). These will be needed to retrieve natural-language labels and other information needed for reuse and display of the data.

The requirements for technically interoperable values in particular can be challenging. Non-IRI values must be in a format that the new application can make use of, for example with date values formatted in a manner that can be accurately interpreted, queried, or displayed by the software. When a MAP specifies that textual or numeric values must conform to a particular SES, ensuring that values do so during metadata creation provides strong support for data reuse.

### Notes

1. Sinopia home page, Linked Data for Production 2 (LD4P2), <https://sinopia.io>.
2. “Metadata Interoperability,” Dublin Core Metadata Initiative, last updated May 6, 2021, [https://www.dublincore.org/resources/glossary/metadata\\_interoperability](https://www.dublincore.org/resources/glossary/metadata_interoperability).

# Components

Theodore Gerontakos

Application profiles (APs) have four major components: (1) the *application*, (2) the *entities* that the AP aims to describe, (3) the *properties* that describe those entities, and (4) the *values* assigned to those properties.

APs and their components have been variously described, most notably in the literature of the Dublin Core Metadata Initiative (DCMI).<sup>1</sup> DCMI's work on APs is an outstanding body of work with which all metadata professionals should be familiar. Many creators of APs, however, are not metadata professionals and will likely find this literature difficult to read. Even many metadata professionals look at the Dublin Core Abstract Model, for example, and wonder, "What does this have to do with the actual work I do?" They file it away and tell themselves, "I'll look at this soon," then don't.

Many people not metadata professionals find themselves tasked with creating an AP. Metadata professionals often advise these people in their efforts, but pointing them toward the DCMI literature is not always the best option. Metadata professionals need to know the DCMI literature; others do not. To play an advisory role, metadata professionals should be aware of what is needed to create APs that function well in the current information landscape and simplify the process for those they advise.

The components of APs have not changed dramatically in the past ten years; the way we write them and the sorts of information they contain have changed. The survey of components below explores familiar AP instruments in the new context, made novel mostly by the introduction of linked-data practices. For those comfortable with AP issues, hopefully we will explore some areas of interest; for others it may serve as a launch into the new context, perhaps even help start or continue updating aging models; however, this exploration most frequently addresses the needs of metadata professionals advising those working on

projects that need APs but lack the expertise, which includes many librarians who are not metadata librarians but are managing projects.

## The Application

An AP is a description of descriptions. It describes to data creators (and others) how to describe something. The application is an application of these descriptions described by the AP. The AP is a set of rules that is completely unnecessary without an application. Strictly speaking, the application is not a component of an AP; rather, it is a determinant, the reason the AP exists.

The use of the word *application* is confusing; there is a distinction between an application for descriptions and the software application that provides an implementation. When we use the term *the application*, we mean the former; when we mean the latter, we use *platform*. Besides the word *application*, more confusion results because many platforms come out of the box with a predetermined application profile. To further complicate matters, we often find that our APs are written for specific platforms. It is not always easy for writers of APs to distinguish between the application and the platform.

All applications have unique needs. The AP is a model that describes descriptions suitable for those unique needs. If the chosen platform comes with an application profile, our job is to understand our metadata needs, choose an appropriate subset of available options, refine the subset, and, if possible, extend it. If we have to create our own AP, we do that by choosing subsets of metadata components that already exist; they just exist outside the chosen platform. In either case, the task of writing an AP is similar: we develop an understanding of the application and design descriptions that will optimize the application's performance.

There is a significant difference between APs written for small projects with relatively low metadata expectations and those for large institution-wide efforts, such as building an institutional repository, where we should expect professional-level data modeling. Small projects can be modeled with short-term visions and informal methods, preferably with some help from metadata professionals. Projects with high expectations involve people across multiple professions, including metadata professionals, who would be deeply involved in modeling the repository. In such cases, professional modeling practices should be scrupulously followed.

Whatever the degree of professionalism needed, the modeling process requires an understanding of the application. Perhaps the fullest description of the AP process is DCMI's Singapore Framework, which can be implemented with varying levels of professionalism.<sup>2</sup> It is an extremely useful recommendation that still has not been widely adopted. Following the Singapore Framework, a metadata application profile, or *description set profile*, is achieved after careful analysis of the application. The instruments for analyzing the application include an evaluation of functional requirements and the creation of a domain model. No specific syntax is required; functional requirements can be a bulleted list of what's required for the application to function well; it can also be written as structured data. Similarly, the domain model could be diagrammed with a pen on an 8½" × 11" sheet of paper; it can also be created using data-modeling tools such as the Unified Modeling Language.<sup>3</sup> In both cases, for most projects, the documentation created at this point may never be reviewed after the application is implemented; what we're creating is a full view of the application that allows us to write our most useful AP. The AP is the documentation we would expect to be used—by data creators and data consumers—after the application is deployed.

When creating models or making recommendations, we often overlook the fact that libraries have worked strenuously to create a data model for library data. In 1997, the *Functional Requirements for Bibliographic Records* (FRBR) was formally released and was updated in 2017 by the *Library Reference Model* (LRM).<sup>4</sup> These are high-level conceptual models that provide a common model for application profiles. They are particularly interested in end users and their use of our applications. They offer a range of instruments for describing the entities end users want to find, identify, select, obtain, or explore (those are the five "user tasks" identified in the model). For most applications and for most platforms, these models are too complex; the most obvious problem is that most of our systems do not differentiate the work, expression, manifestation, and item entities. Nevertheless, the models at the very least can provide a solid foundation for AP

development. The LRM can serve as the starting point for entities in our domain model, as well as for entity attributes, the entities' relationships with other entities, and their alignment with the user tasks.

LRM is a rich source of AP components and is the basis of our current cataloging code, Resource Description and Access (RDA). Both can be used as the basis for creating an overall model, or at least an understanding, of our application, no matter how small the project. This can include evaluating functional requirements, enumerating use cases, creating a domain model that diagrams the things we are describing and how they are related, and consulting already-existing models for the components that will populate our APs. There is no better place to start the AP *process* than an overall evaluation of the application.

## Entities

For many of us, including metadata professionals, the word *entities* is relatively new as an everyday term (despite its use over twenty years ago in FRBR). It refers to the things our applications describe: people, places—everything; concepts (such as subjects); information resources; even knowledge.

We need to identify the things we are describing—that's obvious. Libraries already describe things. The most common things we describe are formats of information resources: a monograph, a map, a sound recording, and so on. We also describe agents, especially in our name authority file. We describe subjects in our subject authority file. Libraries are well acquainted with describing things, just not in a manner fit for the 2020s. There have been several changes in the way things are identified, isolated, described, and brought into relation.

The major reason for these changes is the widespread adoption of linked-data practices. We want to assign Internationalized Resource Identifiers (IRIs) to things. The IRIs act as surrogates for the thing itself. We want to add descriptions of each thing, especially specifying the type of thing, and get the thing, or thing-surrogate, and its descriptions in the web, directly accessible as structured data and not mediated, for example, by a splash page or represented by a record in a library catalog.<sup>5</sup> We want to give our things a web identity or, more precisely, a Semantic Web identity, allowing Semantic Web processors to recognize the thing as a discrete entity.

Everything can be "entified" for the Semantic Web. Not only is the information resource now an entity complex (work/expression/manifestation/item), but it is also related to many other entities. Obvious related entities are the people-entities who produce an information resource, such as the author, the publisher,



and so on; less obvious entities include the title or even the intended audience as things in themselves, things that can be described, identified by an IRI, and assigned a type. Once identified, the entities can then be related using properties.

Application models, especially a domain model, should reveal all types of entities for which our application needs to assign IRIs. Will we be assigning IRIs to works? Expressions? Agents? Keywords? Our APs should describe how to create full descriptions of all things to which we assign IRIs. For all things identified as part of the application but for which we're not assigning IRIs, our APs should identify who has assigned IRIs for those things and how to access those IRIs as well as any additional information about the things.

Unfortunately, most of our current platforms are not optimized for working with IRIs. Some do not create, or “mint,” IRIs. Some do not recognize IRIs as actionable links. Some cannot follow an IRI and retrieve external data using that IRI. Then how many systems recognize entities—such as work/expression/manifestation—or allow for entity creation? Very few! This leads to a range of social problems, from

administrators reluctant to invest in practices that show little return on investment, to colleagues who are openly hostile to linked data. So what are some responses we can adopt now, while in this transition, especially in regard to modeling entities?

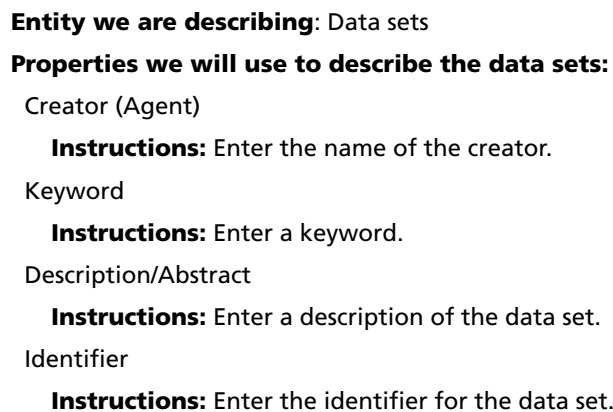
One response is to do little or nothing; just keep creating records. This is not necessarily a poor approach. It represents a confidence that we can give our entities web identities sometime in the future. Our APs in this case look like APs we've been creating the past twenty years. There are very few entities. For example, if we were creating records for data sets, we would likely have only one entity: data sets (see figure 3.1).

If we were describing multiple entities, we would create something that looks more like figure 3.2.

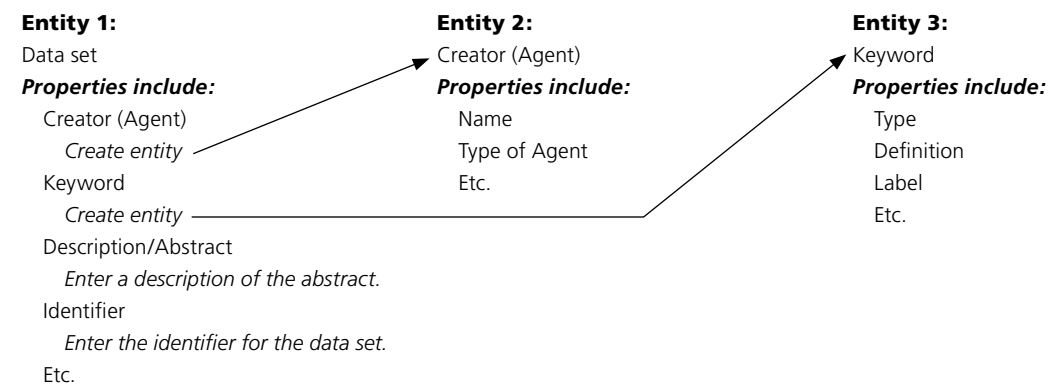
When records are being created, the burden of the AP becomes the description of properties and values, usually for a single entity; in addition, the values will likely be text strings, and the creator of the AP will want to carefully select sources of values. Use of widely adopted standards, best practices, and high-quality well-defined (in the AP) metadata will help ensure future entification. That should not be a revelation: high-quality metadata, using reliable sources for values, has been recommended as a solution to many problems for many decades.

Another response is to start inserting IRIs for entities into our records. This is what libraries are doing that follow the PCC's URIs in MARC Pilot.<sup>6</sup> This can be done in many environments. It does require more work when entering metadata, and the return on investment is not immediately apparent, so it can lead to some resistance. This practice, like the do-nothing approach, represents a confidence that entification and Semantic Web identities will be relatively easily accomplished in the future. The APs we write today would, in this case, need to describe a method for entering entity IRIs (see figure 3.3).

Because many of our systems are not well-suited at present for managing entities, the problem of entities



**Figure 3.1**  
Imaginary AP for creating records describing data sets



**Figure 3.2**  
Imaginary AP for creating data describing multiple entities associated with data sets

**Entity we are describing:** Data sets

**Properties we will use to describe the data sets:**

Creator (Agent) 1

**Instructions:** Enter the name of creator 1 as it appears in English in Wikidata.

IRI for creator 1

**Instructions:** Enter the Wikidata IRI for creator 1.

Creator (Agent) 2

**Instructions:** Enter the name of creator 2 as it appears in English in Wikidata.

IRI for creator 2

**Instructions:** Enter the Wikidata IRI for creator 2.

Keyword 1

**Instructions:** Enter a keyword as it appears in LCSH.

Keyword 1 IRI

**Instructions:** Enter the LCSH IRI for keyword 1.

Keyword 2

**Instructions:** Enter a keyword as it appears in LCSH.

Keyword 2 IRI

**Instructions:** Enter the LCSH IRI for keyword 2.

Description/Abstract

**Instructions:** Enter a description of the data set.

Identifier

**Instructions:** Enter the identifier for the data set.

---

**Figure 3.3**

Imaginary AP for creating records including IRIs for selected entities

can be seen as a future problem. Another thing to do now is to get involved in projects and initiatives that incorporate the new models. This is an excellent way to stay current with many practices, including new ways of writing APs. The projects under “Linked Data for Production,” featuring the linked-data platform Sinopia, are a good example.<sup>7</sup> PCC has had several groups that are working in this arena.<sup>8</sup> A related approach is to maintain, especially in academic libraries, a culture of exploration and research. We should avail ourselves of current information about the benefits and pain points of linked data. Another possibility, something more concrete, is to form an understanding of RDA and LRM as RDF models and start writing application profiles for RDA entities that can be used in actual practice. This last suggestion can apply to other ontologies beyond RDA; many organizations, for example, are writing profiles using the BIBFRAME ontology. This sort of current practice will hopefully ease the difficulty of future entification projects.

Having moved from creating records to creating data signals a significant change in what we do and

creates uncertainty in our APs about how to handle entities. At present, we know we need to continue identifying the things we plan to describe and, at an early stage in the modeling process, establish a general idea of how to describe them. In the new context, the most pressing problem revolves around IRIs and (1) identifying the entities for which we create IRIs and (2) finding already-existing IRIs for the other entities in our data; however, the implications are much broader. Exactly what metadata is, is up for grabs. Our *records* describe information resources (primarily manifestations), agents, and subjects as discrete things, then include textual references to lots of related things. Now when we create data, the list of things we describe, or seek descriptions for, has grown. Our direction is toward the creation of, or linking to, data describing many types of entities and bringing them into relation. In other words, we’re now creating first class data and not only metadata records. Perhaps soon we will not call ourselves metadata professionals but library data professionals?

## Properties

Identifying properties and the correct way to use them is arguably the main purpose of an AP. Most APs are a list of properties and the rules for using them. Although few people look at APs, almost all our users are acquainted with properties. They've been in use for a really long time. In recent times past, we have called them by other names: *field name*, *data element*, *attribute*, *predicate*. Here we call them *properties* and assume a general understanding.

We also favor a current trend in thinking of properties as relationships; specifically, relationships between entities or, more expansively, between resources, because the value of the property may be a literal and is not necessarily an entity. To further complicate matters, LRM and RDA, libraries' primary data models, distinguish relationship-properties from attribute-properties.<sup>9</sup> Nevertheless we remain content to consider the property a relationship between two resources or, otherwise stated, two nodes.

In application profiles, we generally do not define properties; that is done elsewhere, most notably in ontologies or in element sets. Sometimes our AP will require a property that apparently is not defined anywhere; in that case, preferred practice is to define the property, publish the definition, then use it in the AP. The AP assembles predefined properties from multiple sources and clarifies their local use. There are not that many areas that APs clarify for properties, usually ten or so. Figure 3.4 shows five commonly seen properties of properties that we see in almost every AP.

These properties of properties are not terribly complex. What is difficult is deciding. Why is a particular property required? That is an organizational problem. In the miniature AP in figure 3.4, the most complex property is usage notes. It could be called by another name, say *input instructions*; these are specific rules on creating values, which can get quite complex. Often, we find our initial rules do not accommodate all cases, which can lead to a labyrinth of rules in our APs. It can be difficult to find a balance between simplicity and complexity.

Other complexities keep AP creation mostly an endeavor of specialists. These include the need to understand the original context of a property. A property's original context is usually a syndetic structure that can get quite complex. For example, the original context may describe a hierarchy of properties; when we climb the hierarchy, we find our property is in a chain of properties intended to describe humans only. If we use that property to describe dogs, our APs will contribute to the creation of lower quality data.

The complexity we will focus on here concerns changes due to the widespread adoption of linked-data practices. Despite those changes, we can view current efforts as a continuation of what we have always done.

**Required:** yes

**Repeatable:** no

**Data Type:** date

**Definition:** date associated with the item

**Usage Notes:** enter the date the item was created; enter the year only

**Figure 3.4**

Common properties of properties

We still need metadata professionals (catalogers) experienced in data creation—people who know good sources of values, understand the complexities of particular fields like a date field, and so on. The same people, however, were educated in the MARC format; now we need similar expertise in RDF *and* in a handful of additional data models (RDA, BIBFRAME, etc.).

The most notable change is that several properties of properties have become commonplace. Sometimes these properties of properties will be explicit in a source ontology, in which case we can just repeat them in the AP for convenience; other times selected properties of properties needed in the local application will not appear in the source but should be included in the AP. These properties of properties include the following:

### IRI

Properties now have IRIs. The IRI should be explicit in the property's original context. We repeat it in the AP for convenience. If a property does not have an associated IRI, we should consider not using it.

### Label

Human-friendly labels are better than IRIs for display purposes, both within the application and in the AP itself. A label may be recommended in the original context, which is the preferred label, but a local application can use a local label.

### Sub-property-of

Sub-property-of would appear in the property's original context, where it is part of the syndetic structure. We cannot change this information in the AP; we can only repeat it. Care should be taken, when assembling the AP, that the hierarchically inherited properties of properties are respected.

### Domain

Domain may or may not be asserted in the property's original context. If it is, we could simply repeat it in

our AP. Alternatively, we could narrow the domain to a subclass. We should not change the class so that it will result in instances that are no longer members of the class asserted in the original source. If the domain is not asserted in the original source, we are free to assert a domain in the AP, if required by the application.<sup>10</sup>

## Range

Range may or may not be asserted in the property's original context. As with domain, we could choose to refine the range for our purposes, but not change it. If the range is not asserted in the original source, we are free to assert a range in the AP, if required by the application.<sup>11</sup>

## Node Type

The node type concerns a property's value (whether it is an IRI, a blank node, or a literal); we'll describe it in the section on values. However, information about values in our APs is usually documented as part of the description of a property. Node type is a new addition to our APs. It may or may not be defined in the original context of the property.

## "Use Values From" or "Lookup"

"Use values from" or "lookup" will likely appear in an AP only. It is more than an instruction to use headings from a controlled source; it states we should search a controlled vocabulary and, if a match is found for our heading, retrieve additional data from the external data set, especially the IRI. It also may include a search across multiple data sets. It involves new practices for us, and we still don't have a commonly used method to represent these practices in our APs.

Another change is that common practices for creating APs, which may become standards, are emerging. What distinguishes these new practices is the adoption of entities: entities can be specifically described, as well as the profile itself in some cases; nevertheless, the heart of these profiles remains the enumeration of properties used to describe a given entity. The Library of Congress led the way by creating the LC BIBFRAME Profiles specification, which describes a way to describe properties of properties, properties of entities, and properties of the profile itself for unlimited resources.<sup>12</sup> This was adopted and refined by the Linked Data for Production 2 project (LD4P2) for use in creating APs in Sinopia. DCMI is also working on new representations of APs, mostly through its DCMI Application Profiles Working Group, which includes some novel ways to describe and utilize properties of properties but in a familiar spreadsheet environment.<sup>13</sup>

RDA requires a narrowing of its immense number of properties for use in specific applications, and APs will do this work; the RDA literature, notably the RDA Toolkit, includes some sample APs, but it is still unclear what the exact structure will be.<sup>14</sup> Whatever the case may be, a shared structure for RDA profiles would be extremely useful, especially if the structure is to be machine-actionable. Hopefully, PCC will contribute to several of these efforts.

All these efforts at the various leading organizations include the new RDF instruments as well as instruments more traditional to our APs, such as a property's cardinality. Even if we still plan only to create an AP that is a list of properties, we would do well to create APs that take these changes into account. AP creation was already a specialist endeavor that, with a little help, could be handed off to a nonspecialist. With the emergence of linked data, even that may prove difficult.

## Values

With few exceptions, properties have values. In APs, the property-value pair is treated as a unit. Sometimes this unit is called a field. We can distinguish, sometimes with difficulty, a property of a property from a property of a value. For example, stating that a property is repeatable is a property of the property; stating that the property uses only terms from the Art and Architecture Thesaurus is a property of the value. Nevertheless, if we are creating a tabular AP, with each row describing a property, the properties of the value are described in the same row as the properties of the property.

Here we will endeavor to discuss the properties of values. Until recently, these properties were almost exclusively ways to create textual values ("strings"). This is still an important part of the properties of values, but with the introduction of linked-data practices, we have not only some new ways of describing string values, but also some new value properties.

*Node type* is a new property of values mentioned above. It is a springboard into many current issues, so we will take a close look at node type, and then, due to space constraints, take only a cursory look at some other properties of values.

A *node* is derived from the domain of graph data modeling; it comes to most library metadata professionals from RDF specifications, as RDF is a graph data model. It's simple: every thing or string is a node joined to another node by a property, also called a "relation," or an "arc." The RDF core structure is often represented as the "triple": (1) a *subject* node that can be described by a property or "arc" called (2) a *predicate* that has a value that is (3) the *object* node. Many of us are familiar with the RDF triple

subject-predicate-object, but most people we collaborate with will not be familiar with graphs and RDF, so talking about nodes may be a little confusing. The triple-subject and the triple-object are nodes.

When we talk about the node type, we are describing the triple-object. Although not all data for which we will want to write APs will be RDF data, it is useful to keep in mind that, when we describe values, we're describing the triple-object, the value of the predicate, the node toward which the arc in the graph is directed.

There are not many node types: IRI, blank node, literal, or some combination of the three. Distinguishing the node type is useful. An IRI and a blank node represent actual things, and things require additional modeling and description. If the node is a literal, then rules for entering the literal should be included in the AP. However, a literal is not a thing. It can be turned into a thing, like the RDA Nomen, but, as a string, it cannot be further described.

Of course an IRI is itself a text string distinguished because it is situated in larger data models. In the context of the World Wide Web, it is an actionable string that follows a particular syntax; the action is that it "dereferences" (the IRI is an IRI "reference" that references a resource on the web). This points to another complexity that makes AP authoring a task of specialists. Our APs are miniature models that require a handful of skills. These miniature models are part of larger models, such as OWL ontologies. Those models themselves are situated in a mega-model, RDF, which provides a common model for Semantic Web data. Then all of this is situated in a super-mega-model, the World Wide Web. Understanding the full stack is more than a full-time endeavor.

Stating that a value is an IRI has easily understood implications for our instance data. Any values to be entered by data creators for a given property should be IRIs; any values seen in a data set by data consumers can be recognized as IRIs. If sound linked-data practices are followed, the IRI represents a thing, and the data consumer should be able to dereference the IRI and retrieve useful information about the thing.

There are nodes that are things but are not represented by IRIs and are not text nodes; we call these *blank nodes* or *bnodes*. They can be described—statements can be made about them—but they are not given an identity on the web. As usual, it is more complicated than that: usually local identifiers are assigned (by any software used to parse the data), but these do not persist beyond the local context.

Stating that a value is a blank node has easily understood implications for our instance data. Any values to be entered by data creators for a given property should be a blank node and should not be assigned an IRI. This is not straightforward for our systems, however; blank nodes require systems that

permit a layered data structure. Blank nodes result in data "nested" in other data. As writers of APs, we are not always at liberty to state that a value should be a blank node. It depends on our model developed for the overall application within the confines of a specific platform (our "implementation model").

The string is the most complicated of the three node types: it can be "structured," "unstructured," or an identifier; it can be "typed"; its language can be stated; it can be from a controlled vocabulary.

*Structured* and *unstructured* are terms taken from RDA to describe literals, but the problem they represent is not just an RDA problem. An unstructured literal is a blob of text entered however, without any rules; it is uncontrolled. A structured literal, on the other hand, follows data-entry rules; for example, the elements of the textual information may need to be entered in a particular order. The rule followed is called a *string encoding scheme* in RDA; elsewhere, most notably in DCMI, it is called a *syntax encoding scheme*. Nowadays it is common to call it an *SES*, and it is another property of values. In our AP, we would somehow state that the value of a given property is a literal and that the literal is either structured or unstructured; if structured, we would specify the SES.

RDA also distinguishes literals that are identifiers. Although these literals are similar to structured values, they are distinguished not only because they always have meaning in a particular external context, but also because they are considered machine-readable. Although not a common feature of APs outside RDA, identifier-literals will be essential to our RDA profiles (when we get around to writing them).

Another node type is the *typed* literal. Usually, in an AP, we state that the node type is a literal and that the data type is a particular data type. Because of the overuse of the word *type*, there is some confusion with this property of values. In this case, we're referring to traditional data type, such as string, date, dateTime, number, integer, and so on. When our instance data follows RDF, our typed literals are most frequently typed using the data type vocabulary in "W3C XML Schema Definition Language Part 2."<sup>15</sup> Our entry in the AP could look something like the fragment in figure 3.5.

When the node type is a literal, there is yet another variation: the literal with its language identified. This presents a particular problem for the AP; we would state that the node type is a literal, then state somehow (it is not a fixture of APs at present) what the language expectations are. It could be that a value must be in a particular language; that is easy to represent in the AP, say in the input instructions. It could be that the language of the value must be explicit in our instance data; that is more difficult: where do we enter that information? Our systems do not customarily allow us to attach a language to a value (see figure

**Property:** <http://purl.org/dc/terms/date>

**Label:** Date

**Node Type:** rdfs:Literal

**Datatype:** xsd:gYear

**Input instructions:** Enter the date created; enter the year only

**Figure 3.5**

Property/values for a date property plus property/values for values, including data type

**Application Profile:**

**Property:** Label

**Node Type:** Literal

**Language:** Greek, Ancient [this is a property of the value and its value]

Display 1:

**Label:** Ἀνδρομάχ

Display 2:

**Label:** Ἀνδρομάχ [Language: Greek, Ancient]

Stored as:

Label=47;

47.labelString: Ἀνδρομάχ

47.labelLanguage: GreekAncient

**Figure 3.6**

Difficulties making language of a value explicit

3.6).

This “meta” problem, which applies also to data type, is a structural complexity inherited partially from RDF. We want our systems to represent a value of a property of a value. These seemingly simple needs cause trouble at many levels. Where does the information go in the AP? Where in the data entry form? Should all values have a language requirement? Nevertheless, we widely acknowledge the importance of language identification and would do well to create data points for language in our APs and demand systems that feature elegant representation of language.

The last type of literal we will consider here is a literal value taken from a controlled vocabulary. The new term for a controlled vocabulary is *vocabulary encoding scheme* or VES. Ideally values from a VES would have a node type IRI, not literal. The result in most systems, unfortunately, would be a value that displays to users as an IRI. Surely users would prefer we enter the literal instead. In this case, the VES would be identified in our AP, but its IRIs would likely not appear in our instance data (see figure 3.7).

As seen above, other properties of values include the SES and the VES; also mentioned above were domain and range, as well as data type, which can

**Property label:** Subject

**Node Type:** literal

**SES:** LCSH at <http://id.loc.gov/authorities/subjects.html>

**Input instructions:** Enter the header exactly as it appears at the top of the page

**Figure 3.7**

AP fragment with instructions to use a string from the VES

be seen as properties of values. Sometimes we see “shape,” which comes to us from RDF validation languages (SHACL and ShEx being the most prominent). Cardinality can be a property of values when a single property allows multiple values separated by a delimiter. Other properties of values might include

- length of the value
- choice of a value from a set
- intersections or unions of value sets
- constant values
- maximum/minimum
- maxInclusive/minInclusive
- base IRIs for IRI values
- patterns that values should follow (like regular expressions).

A lot of this may be present in the ontologies we use as sources for our APs, or we may add it to the APs ourselves. If it is present in the ontology, we would not want to contradict anything in the ontology.

## Notes

1. Dublin Core Metadata Initiative, <https://dublincore.org/>.
2. Mikael Nilsson, Tom Baker, and Pete Johnston, “The Singapore Framework for Dublin Core Application Profiles,” Dublin Core Metadata Initiative, January 14, 2008, <https://dublincore.org/specifications/dublin-core/singapore-framework>.
3. Unified Modeling Language, <https://www.uml.org/>.
4. IFLA Study Group on the Functional Requirements for Bibliographic Records, *Functional Requirements for Bibliographic Records*, IFLA Series on Bibliographic Control 19 (Munich, Germany: K. G. Saur Verlag, 1998, last updated February 2009) <https://www.ifla.org/publications/functional-requirements-for-bibliographic-records>; Pat Riva, Patrick Le Bœuf, and Maja Žumer, *IFLA Library Reference Model: A Conceptual Model for Bibliographic Information* (The Hague, Netherlands: International Federation of Library Associations and Institutions, August 2017, last updated December 2017), <https://www.ifla.org/publications/node/11412>.
5. Tom Heath and Christian Bizer, *Linked Data: Evolving the Web into a Global Data Space* (San Rafael, CA:

- Morgan & Claypool, 2011), chap. 3, <http://linkeddatabook.com/editions/1.0>.
6. "PCC URIs in MARC Pilot," Program for Cooperative Cataloging, Library of Congress, last updated October 24, 2019, <https://www.loc.gov/aba/pcc/pilots/URIs-in-MARC-Pilot.html>.
  7. Sinopia home page, Linked Data for Production 2 (LD4P2), <https://sinopia.io>.
  8. "Task Groups," Program for Cooperative Cataloging, Library of Congress, <https://www.loc.gov/aba/pcc/taskgroup/task-groups.html>.
  9. Riva, Le Bœuf, and Žumer, *IFLA Library Reference Model*, 17.
  10. Dan Brickley and R. V. Guha, eds., "RDF Schema 1.1," section 3.2, W3C Recommendation, World Wide Web Consortium, February 25, 2014, [https://www.w3.org/TR/2014/REC-rdf-schema-20140225/#ch\\_domain](https://www.w3.org/TR/2014/REC-rdf-schema-20140225/#ch_domain).
  11. Dan Brickley and R. V. Guha, eds., "RDF Schema 1.1," section 3.1, W3C Recommendation, World Wide Web Consortium, February 25, 2014, [https://www.w3.org/TR/2014/REC-rdf-schema-20140225/#ch\\_range](https://www.w3.org/TR/2014/REC-rdf-schema-20140225/#ch_range).
  12. "BIBFRAME Profiles: Introduction and Specification," draft for public review, Library of Congress, May 5, 2014, <https://www.loc.gov/bibframe/docs/bibframe-profiles.html>.
  13. Dublin Core Metadata Initiative, "dcmi/dctap," GitHub, <https://github.com/dcmi/dctap>.
  14. "Application profiles," RDA Toolkit, [https://access.rdatoolkit.org/Guidance/Index?externalId=en-US\\_ala-591ca278-2807-399b-9530-6b44171e6ccc](https://access.rdatoolkit.org/Guidance/Index?externalId=en-US_ala-591ca278-2807-399b-9530-6b44171e6ccc).
  15. David Peterson, Shudi (Sandy) Gao, Ashok Malhotra, C. M. Sperberg-McQueen, and Henry S. Thompson, eds., "W3C XML Schema Definition Language (XSD) 1.1 Part 2: Datatypes," W3C Recommendation, World Wide Web Consortium, April 5, 2012, <https://www.w3.org/TR/xmlschema11-2>.

# Sources

Theodore Gerontakos

An essential part of the AP process is to determine the sources of the components we mix and match. Although it's true that linked-data practices and other technologies have changed our work, there has been an increase in linked-data-ready sources available. The quantity of tools available at the Library of Congress Linked Data Service alone is remarkable, not to mention all the other vocabularies and ontologies available as linked-data resources.

*Library of Congress Linked Data Service*  
<https://id.loc.gov/>

In this chapter on sources, we'll look at four sources of AP components: ontologies, schemas, vocabulary encoding schemes, and syntax encoding schemes, then conclude with some ideas on how to find them.

## Ontologies

Ontologies have emerged as our richest source of AP components. Creating an ontology is difficult, as they often model total knowledge domains. Modeling library data alone is an enormous undertaking that requires many entities and properties. Once complete, however, many ontologies are relatively easy to use for assembling APs. They are essentially lists, with attendant descriptions, of classes and properties. The classes are types for our entities; our entities are instances of a given class. The properties relate the entities to other entities.

The ontologies that concern us here are RDF ontologies. These can be expressed using classes and properties specified for creating descriptions of classes and properties. The core instruments for creating RDF

ontologies are found in RDF Schema 1.1 (RDFS).<sup>1</sup> We find the classes `Class` and `Property` in RDFS (although `Property` is technically in the RDF namespace, not RDFS) to classify instances of those classes. We also find properties such as `domain`, `range`, `subPropertyOf`, and `subClassOf`. RDFS is a small but essential ontology for creating ontologies.

The Web Ontology Language (OWL) is a family of ontology languages that incorporates and extends all of RDFS and vastly increases its expressive power.<sup>2</sup> Using OWL, classes, properties, and complex values of properties can be described using instruments based in formal logic. Most of the ontologies we use to create library data do not dive too deeply into OWL constructs, but, rather, use elements of OWL when they are useful.

An actual RDF ontology is serialized as RDF and usually can be accessed in one of many possible RDF serializations (RDF/XML, Turtle, N-Triples, etc.). In addition, it has become commonplace to make the ontology viewable as a human-friendly HTML page. All modes of access should be available over the World Wide Web. The BIBFRAME ontology is a good example: it can be accessed in many different flavors of RDF using a download link embedded in a web page; it can also be viewed in its totality as an HTML page, with some prefatory material on top, then the class list, followed by the property list, all items in each list hyperlinks to class and property descriptions.<sup>3</sup>

In the previous chapter, we saw how using properties can be complex because, in their original context, they are part of a syndetic structure. Ontologies require the same level of expertise to use accurately. The classes and properties lists are taxonomies whose syndetic structure should be analyzed before they are used. Although ontologies are model-centric conceptualizations that help us *interpret* data, not data-centric rules that validate data—ontologies are not constraints—if we do not create data that agrees with



our source ontologies, the truth value of our instance data will decrease. The difference between the closed world of constraints and the open world of ontologies is subtle and difficult to discern. Sometimes it seems best to understand the concepts in an ontology as if they were constraints.

Currently the primary model for library data is IFLA's *Library Reference Model* (LRM).<sup>4</sup> This is a high-level conceptual model that consolidates three earlier IFLA conceptual models, FRBR, FRAD, and FRSAD. It is intended to be a broad logical structure that more detailed ontologies align with. RDA can be considered an ontology that extends and aligns with LRM.<sup>5</sup> RDA, along with BIBFRAME, has become one of two major ontologies for describing library data.

RDA is a complete ontology for describing thirteen entities crucial to library data, including the famous Group 1 entities inherited from FRBR, namely Work, Expression, Manifestation, and Item. RDA's most striking feature, at first glance, is a large quantity of properties. The resulting instance data created using the RDA ontology, with relatively few classes and many properties, is not layered and nested but quite flat, and so is easily rendered using RDF triples (although RDA does not assume RDA data will be expressed as RDF). In addition to abundant properties, RDA provides detailed instructions on forming values. The large quantity of properties also includes precise relationships between entities, making RDA the only ontology to date that can express the modeling of bibliographic relationships that is one of the great achievements in library science over the past thirty years.

Organizations that intend to use RDA are advised to create APs for its use. The magnitude of properties needs to be narrowed for specific applications, and there are multiple options for forming values and for creating new entities that require rulings (for example, is the item in hand a new work or an expression of an already-existing work?). Exactly what these profiles will look like is undetermined. One of the locations for viewing RDA, the RDA Toolkit, displays some human-readable APs that offer a clue.<sup>6</sup>

BIBFRAME is the second of our two major ontologies. BIBFRAME is not aligned with LRM and lacks the expressive power of RDA when describing relationships. However, BIBFRAME features a rich taxonomy of classes and has de facto become the more widely adopted ontology. Instance data created using BIBFRAME is deeply layered and nested and can be difficult to process and query. It does not model bibliographic relationships in detail, which creates a significant loss of data when converting RDA data to BIBFRAME data. In addition, BIBFRAME uses a different model for describing products of intellectual and artistic endeavors than RDA, eschewing the RDA entity Expression, which has created an incompatibility

between the two models. Although some adopters of BIBFRAME aspire to form values in BIBFRAME using RDA rules, RDA data is more accurately rendered using the RDA ontology.

The Linked Data for Production 2 (LD4P2) project participants created APs for use in the Sinopia platform.<sup>7</sup> These APs were authored using the LC BIBFRAME Profiles specification.<sup>8</sup> They can be regarded as state-of-the-art APs. Almost all the APs focus on the implementation of BIBFRAME classes and properties; one notable exception is the University of Washington contribution, which focuses on implementing RDA classes and properties.<sup>9</sup>

Whether using BIBFRAME, RDA, or other ontologies, rarely is a single ontology entirely sufficient to describe our resources. It is common to combine properties and classes from multiple ontologies using an AP, as well as to annotate and refine the uses of those classes and properties. In addition to multiple ontologies, ontology extensions are useful sources for APs; these are mini-ontologies that provide suitable classes and properties for applications not covered by the parent ontology. In one of the original Linked Data for Production initiatives (LD4P), several ontology extensions were written to extend BIBFRAME.<sup>10</sup>

There are many other ontologies that are useful for creating library data. These include the following:

- General
  - Dublin Core
  - Schema.org
  - CIDOC-CRM
  - MODS
  - DPLA
  - Europeana
- Authority data: MADS
- Preservation metadata: PREMIS
- Taxonomies and thesauruses: SKOS
- People
  - FOAF
  - VIVO
- Data sets
  - VoID
  - DCAT

## Schemas

*Schema* is a word used frequently when discussing metadata. Here we are referring to a metadata element set. As usual, however, it is a little more complicated than that. Metadata schemas are something we often referred to in the XML era.<sup>11</sup> Usually *metadata schema* means a complete element set, commonly presented as a document intended for human consumption (an HTML page, a PDF, etc.) backed up by code written

in a schema language (most often XML Schema). The code's main purpose is to validate, or constrain, the data.

Many monumental schemas have been produced, including EAD, METS, MODS, PREMIS, DDI, and many more. These schemas aspire to meet all an application's descriptive needs, and instance metadata can be stored as a machine-actionable XML document or document-like object. Machine actions, or processing, focus on validation, but also could include additional processes such as conversion to other metadata schemas.

Application profiles emerged in the XML era, mostly as an attempt to supplement general metadata schemas such as Dublin Core; however, sometimes even the comprehensive schemas require extensions using elements from other schemas. In most cases, these APs are human-readable documents with a list of properties, the properties of the properties, their source, and sometimes mappings to other schemas. XML tools are powerful and permit metadata professionals to go far beyond human-readable documents; nevertheless, the common practice was to remain practical and meet local needs with appropriate effort. APs created in spreadsheets for immediate local use were common.

These practices persist. It is still common for projects to create only human-readable APs, and we still see instance metadata stored in an XML document, often constrained by an XML schema that accompanies an element set—or even an XML schema based on an AP. Although linked-data practices prefer a totalized machine-actionability and RDF serialization with dereferencing capability in the web, XML-era instance data is still abundant and useful. It can even be used to derive RDF data. As stated, XML tools are powerful, and XML data should persist well into the future.

Over the years many element sets have been rebuilt as ontologies, which is no small feat. The element sets are independent resources with an internal consistency; to rebuild them as stand-alone ontologies, all the properties need to be defined, syndetic structure retained, entities identified—it's a lot of work. In the meantime, many schemas can still be used advantageously, and APs can still be assembled and backed up with local validation code. If nothing else, schemas can be a rich source of ideas for creating APs and other local models. As we've seen above, we continue to describe and borrow elements the same way we have for many years. We've just added a few new practices to improve data integration with the web in the 2020s.

Here is a list of some well-known schemas: EAD, MARCXML, MODS, VRA Core, Dublin Core, OAI-PMH, OAI-ORE, METS, PREMIS, XMP, EXIF, IPTC, MPEG-7, PBCore, DDI, Darwin Core, ONIX, TEI, and MIX.

## VES

Librarians are well-acquainted with vocabulary encoding schemes (VESs), also known as value encoding schemes: they are controlled vocabularies—thesauri, taxonomies, classification schemes, subject headings lists, and so on. They are sources for values. The term was adopted at DCMI to refer to complete lists: the complete thesaurus, taxonomy, and so on. This was necessary because individual terms did not have an IRI, and the source of each heading needed to be recorded in our instance data, preferably using an IRI or URL. With the advent of a functioning linked-data infrastructure, many VESs have assigned IRIs to each heading and have described each entry as a discrete resource; reference to the full thesaurus is not as important as it once was in our instance data. Still, VES identification remains an important feature of APs.

Ideally a term from a VES would be represented in our instance data as an IRI. RDA, however, allows terms from VESs to be recorded in our instance data using any of the RDA recording methods (structured, unstructured, identifier, IRI). The VES in this case is seen as a source of values regardless of the node type expected for a value.

RDA offers a range of RDA value vocabularies, or VESs, in the RDA Toolkit and the RDA Registry.<sup>12</sup> The Library of Congress offers a tremendous range of VESs at its Linked Data Service. There are many well-used, widely adopted value vocabularies.

Value vocabularies were considered a sensible place to start creating our linked-data infrastructure. In the 2011 Library Linked Data Incubator Group Final Report, they were singled out as “low-hanging fruit” for conversion to linked data.<sup>13</sup> As a result, there are many value vocabularies that are excellent sources of values, whether they're used for the literals, the identifiers, or the IRIs.

## SES

A syntax encoding scheme (SES) is a set of rules for constructing a literal value. As a source for APs, it is less a source of values and more a set of rules for creating a text string.

Many SESs exist. The W3C Date and Time Formats (W3CDTF) is an example.<sup>14</sup> It is a set of rules on how to construct dates. There are rules on how to represent languages, geographic coordinates, access points for information resources, and many more.

DCMI considers an SES a data type.<sup>15</sup> As such, an SES-structured value can be rendered in RDF data as a typed literal, described above in the “Values” section of chapter 3. To facilitate this, Dublin Core Metadata

Terms currently lists twelve SESs and assigns IRIs. Thus for a date structured using W3CDTF, an RDF object node could be entered as follows: “1997-07-07”^^<<http://purl.org/dc/terms/W3CDTF>>.

RDA calls an SES a “string encoding scheme.” The RDA definition is difficult to understand: “A set of string values and an associated set of rules that describe a mapping between that set of strings and a value of an element.”<sup>16</sup> Here we maintain the phrase “set of rules” and take comfort that we can think of an SES as a data type.

In our APs, we should specify the source of the set of rules. If convenient, we can simply repeat the rules in the AP. However, in most cases this would unnecessarily overburden our AP with excessive detail when we could simply reference the SES. Again, it is a judgment we need to make to balance simplicity and complexity.

## Finding Sources

When we want to create our AP or extend a schema or ontology, how do we find these sources of properties, values, and classes?

In most cases, metadata professionals are just familiar with available resources or find them by querying the web.

An alternative is to consult lists of resources available; librarians are well-known for their lists of resources, and lists of resources for writing APs are available (sometimes embedded in more generalized lists). Here are a few:

- “Semantic Web and Linked Data,” UCLA Library, [https://guides.library.ucla.edu/semantic-web/semantic\\_web\\_vocabularies](https://guides.library.ucla.edu/semantic-web/semantic_web_vocabularies).
- “Lists of Ontologies,” World Wide Web Consortium, last updated December 13, 2013, [https://www.w3.org/wiki/Lists\\_of\\_ontologies](https://www.w3.org/wiki/Lists_of_ontologies).
- “Metadata for Data Management: A Tutorial,” UNC University Libraries, last updated January 25, 2021, <https://guides.lib.unc.edu/metadata/standards>.
- “Metadata Standard,” Wikipedia, last updated April 15, 2021, [https://en.wikipedia.org/wiki/Metadata\\_standard](https://en.wikipedia.org/wiki/Metadata_standard).

The preferred method is to find sources of properties, classes, or values directly on the web, as if we were searching a metadata registry.<sup>17</sup> This is not yet a mainstream practice, but there are some implementations, most notably Linked Open Vocabularies (LOV).<sup>18</sup> Other examples include BioPortal, Biblioport, and a taxonomy search called TaxoBank.<sup>19</sup>

The bad news is that there is no easy way to find components for an AP. This is another reason AP authoring is performed with the assistance of a specialist.

## Notes

1. Dan Brickley and R. V. Guha, eds., “RDF Schema 1.1,” section 3.2, W3C Recommendation, World Wide Web Consortium, February 25, 2014, <https://www.w3.org/TR/2014/REC-rdf-schema-20140225>.
2. OWL Working Group, “OWL: Web Ontology Language (OWL),” World Wide Web Consortium, December 11, 2012, <https://www.w3.org/OWL/>.
3. “BIBFRAME Ontology,” v. 2.0.1, Library of Congress, <https://id.loc.gov/ontologies/bibframe.html>.
4. Pat Riva, Patrick Le Bœuf, and Maja Žumer, IFLA Library Reference Model: A Conceptual Model for Bibliographic Information (The Hague, Netherlands: International Federation of Library Associations and Institutions, August 2017, last updated December 2017), <https://www.ifla.org/publications/node/11412>. The LRM is also expressed as RDF at <https://www.iflstandards.info/lrm/lrmer.html>.
5. RDA Toolkit, <https://access.rdatoolkit.org>.
6. “Application profiles,” RDA Toolkit, [https://access.rdatoolkit.org/Guidance/Index?externalId=en-US\\_ala-591ca278-2807-399b-9530-6b44171e6ccc](https://access.rdatoolkit.org/Guidance/Index?externalId=en-US_ala-591ca278-2807-399b-9530-6b44171e6ccc).
7. “Profiles on GitHub,” LD4P2 Linked Data for Production: Pathway to Implementation,” last updated March 31, 2019, <https://wiki.lyrasis.org/display/LD4P2/Profiles+on+GitHub>.
8. “BIBFRAME Profiles: Introduction and Specification,” draft for public review, Library of Congress, May 5, 2014, <https://www.loc.gov/bibframe/docs/bibframe-profiles.html>.
9. Linked Data for Production, “LD4P/sinopia sample profiles,” GitHub, last updated June 22, 2020, [https://github.com/LD4P/sinopia\\_sample\\_profiles/tree/master/cohort-profiles/university-of-washington](https://github.com/LD4P/sinopia_sample_profiles/tree/master/cohort-profiles/university-of-washington).
10. “LD4P Outputs,” Linked Data for Production, last updated August 5, 2018, <https://wiki.lyrasis.org/display/LD4P/LD4P+Outputs>.
11. Use of XML in libraries flourished ca. 2000–2017—the “XML Era”—but it is still very much alive. The XML tool suite is an outstanding tool set that libraries should continue to use well into the future.
12. “RDA Registry,” RDA Steering Committee, in association with ALA Digital Reference, <https://www.rdaregistry.info/>.
13. W3C Library Linked Data Incubator Group, “4.1.1 Identify Sets of Data as Possible Candidates for Early Exposure as Linked Data,” *Library Linked Data Incubator Group Final Report*, World Wide Web Consortium, October 25, 2011, [https://www.w3.org/2005/Incubator/lld/XGR-lld-20111025/#Identify\\_sets\\_of\\_data\\_as\\_possible\\_candidates\\_for\\_early\\_exposure\\_as\\_Linked\\_Data](https://www.w3.org/2005/Incubator/lld/XGR-lld-20111025/#Identify_sets_of_data_as_possible_candidates_for_early_exposure_as_Linked_Data).
14. Misha Wolf and Charles Wicksteed, “Date and Time Formats,” World Wide Web Consortium, September

- 15, 1997, <https://www.w3.org/TR/NOTE-datetime>.
15. "Syntax Encoding Scheme," Glossary, Dublin Core Metadata Initiative, last updated May 6, 2021, <https://www.dublincore.org/resources/glossary/#Syntax%20Encoding%20Scheme>.
16. "Glossary," RDA Toolkit, <https://access.rdatoolkit.org/Glossary>.
17. Such as, for example, the Open Metadata Registry at <http://metadataregistry.org/>.
18. Linked Open Vocabularies (LOV), <https://lov.linkeddata.es/dataset/lov/>.
19. BioPortal, <https://bioportal.bioontology.org/>; Bibliportal, <http://biblio.ontoportal.org/>; TaxoBank Terminology Registry, <http://www.taxobank.org/>.

# Presentations

Benjamin Riesenber

Selection of a presentation format for a MAP will depend on the local needs and applications that it is designed to support. A MAP for use in the creation or transcription of metadata values by catalogers and other staff should provide human-readable guidance, perhaps including detailed instructions for creating, selecting, and formatting values. When using metadata-creation tools, a machine-readable MAP can pass customizations and constraints such as value data type, cardinality, and VESs for the selection of values to a data-entry form. In these cases an additional human-readable format may also be helpful during descriptive work. For some applications, metadata values for digital objects such as file format, size, and checksum may be generated or extracted based on the specifications in a machine-readable MAP, with no need for human-readable guidance at the point of metadata creation.

## Natural Language MAPs

To successfully implement any MAP, the requirements for a metadata model will need to be provided in human-readable format at some point, even if only as a set of specifications provided to software developers. A large portion of existing MAPs have been designed expressly for reading and use by humans. This makes sense in the context of libraries, where metadata is often created by staff who need a comprehensive set of instructions to perform their work.

The Metadata Application Profile Clearinghouse Project reflects the prevalence of human-readable MAP presentations. This online resource provides access to MAPs submitted by a variety of organizations, with a focus on those created for use in describing collections in digital repositories. While the Clearinghouse provides only a very small cross section of

MAPs, the prevalence of human-readable formats is remarkable: all eighteen organizations that have submitted MAPs to date use one or more human-readable versions, and only one of these has provided versions for machine processing.<sup>1</sup>

Many questions arise for metadata creators in the process of describing information resources. Content standards for library cataloging are complex, and the need for guidance makes it unsurprising that human-readable MAP formats are the most commonly used in libraries, as they can provide the rules for generating metadata alongside examples of properly formed values and other relevant information. The popularity of human-readable MAP formats means that there are abundant examples in this category.

## The BIBCO Standard Record

The BIBCO Standard Record (BSR) RDA Metadata Application Profile is a baseline set of elements applicable to the description of a wide variety of resource formats commonly collected by libraries. It was created by the Library of Congress Program for Cooperative Cataloging's (PCC) BIBCO program in support of its work to improve the quality of bibliographic description and support efficient cataloging. The ninety-six (as of fiscal year 2020) BIBCO member institutions and “funnels”—groups of libraries or individual catalogers that work together to contribute records—as well as other institutions that have adopted the standard voluntarily, use it as fundamental guidance for creating catalog records.<sup>2</sup>

The BSR is presented in a human-readable, primarily tabular format and provides information including the following:

- headings enumerating physical and intellectual entities for description—for example, “Identifying

Works & Expressions,” “Identifying Manifestations & Items,” and “Describing Carriers”

- properties (referred to in the BSR as “elements”) grouped under these headings and others

For each element, the following information may be given:

- for each element taken from the Resource Description and Access (RDA) set of properties, a link to detailed online guidance where available
- notes including information about specific formats for which the element is recommended or required, additional instructions for recording values, and information about where to find additional guidance

### Wikidata Application Profiles

Wikidata is a free and open linked-data knowledge base and, like the ubiquitous Wikipedia platform, a project of the nonprofit Wikimedia Foundation. As evidenced by a Library of Congress PCC pilot project launched in 2020 to explore its use, with more than seventy institutional pilot participants including many college and university libraries in the United States, its visibility as both an editing tool and publishing platform for bibliographic description is increasing.<sup>3</sup>

Many communities are developing and publishing human-readable MAPs to guide resource description in the Wikidata platform, including MAPs for use describing books, periodicals, and video games. The MAP published by WikiProject Books provides an enumeration of several conceptual entities that may be used to create bibliographic descriptions, including written works; versions, editions, or translations; exemplars; and manuscripts. Properties recommended for use with each are provided in a tabular format. For each property, information including the following is provided:<sup>4</sup>

- a property label and an ID linking to a full definition
- a required data type for values
- a property description providing brief guidance for entering values
- examples of use, linking to existing item descriptions

### Encoded MAPs

The continuing evolution of software tools used to create, manage, and serve metadata provides increasing opportunities for MAPs to integrate directly with them. As a result, growing numbers of encoded,

machine-readable MAPs are available as examples. Many of these MAPs have been developed for RDF applications.

### BIBFRAME Profiles

BIBFRAME Profiles are MAPs for describing specific kinds of entities: instances of the resource classes defined in the BIBFRAME RDF vocabulary.<sup>5</sup> They encode definitions in a way that can be interpreted by a specific software tool, the BIBFRAME Editor, which uses the encoded information to generate data-entry forms. The form input provided by a user is combined with information from the Profile to create an RDF metadata instance.

For example, the class of each entity described in a generated metadata instance is taken from the profile. While the Editor uses labels for properties to make the data-entry form more readable for catalogers, properties in the instance are identified by the IRIs specified for them in the profile and paired with the value or values entered by users. This combining of user input with information from a MAP may be clearer if we look specifically at the information that can be included in a BIBFRAME Profile:

- Basic metadata about the profile itself, including an author, date, title, and description; this assists users in managing profiles and selecting them for use.
- A label, IRI, and guiding statement for each entity type that can be described using the profile.
- A set of properties, each identified with an IRI, that may be used for each entity type.

For each property, the following information may be specified:

- whether a value is required
- whether multiple values can be entered
- whether the interface will prompt entry of a literal value or an IRI selected using a query interface in the editor
- a VES from which IRI values may be selected
- a data type to be assigned to literal values
- a default value

The Library of Congress has developed the BIBFRAME Profile Editor software package, which is accessible online and can be used to create, view, edit, and export BIBFRAME Profiles.<sup>6</sup>

### Sinopia Resource Templates

Like the BIBFRAME Editor, the Sinopia Linked Data Editor is a tool for creating linked-data resource descriptions and requires encoded information to

structure data-entry forms and generate data from user input. These MAPs were originally encoded using a structure very similar to that of a BIBFRAME Profile. Sets of entity types for description, and properties and value constraints for use with each, were defined and encoded as a single file. Following recent development of the Editor software package, this information is now organized by single entity types, resulting in a resource template that includes a single entity definition, properties for use with the defined resource type, and value constraints for each property. These resource templates provide much the same information as that given for an entity type in a BIBFRAME Profile and also include basic metadata about the templates themselves.

Additionally, Sinopia resource templates are themselves sets of RDF triples, constructed using terms from the Sinopia Vocabulary, published in 2020.<sup>7</sup> Because resource templates are constructed as RDF graphs in the same manner as descriptive metadata sets, they may, like metadata sets, be created and edited using the Linked Data Editor.

## Hybrid MAPs

MAP information is always required in human-readable form at some point during the implementation process. In many cases human-readable formats play the primary role, providing information essential to knowledge workers creating resource descriptions. At the same time, new and evolving software platforms for metadata creation are providing more opportunities for encoded MAPs to integrate with them directly, enforcing requirements and value constraints more consistently by integrating with processes for metadata creation and management.

But this opportunity may also mean there is a need to create and maintain two separate MAP formats for a single implementation—one to provide guidance to humans, and one to pass information to software. The ability to use a single MAP format for both humans and machines would be ideal in such cases and would avoid duplicating work. In cases where a single format cannot meet all needs, implementers may benefit from tools to convert back and forth so that updates and changes need be made in only one place.

### One MAP for Two Purposes

Any MAP encoded for machine processing can be considered human-readable. Realistically, however, this is limited to people with knowledge in multiple areas, including the MAP's domain model, its data serialization and structure, and the systems that will process it. While presenting encoded MAPs as reference for human users may be efficient, the required knowledge

is a significant barrier to readability. This challenge may be lessened by using a machine-readable format that is relatively simple and accessible.

#### VALIDATION CODE AS MAP

A number of coding languages exist for the purpose of validating data; one well-known example is the XML Schema Definition Language. Languages such as this allow users to encode requirements for metadata instances in a precise way, and this code can be processed along with metadata in order to accurately and efficiently identify portions of an instance that don't conform to requirements.

Chapter 2 discussed the development and implementation of MAPs primarily in terms of work that takes place prior to or during metadata creation. Validation code can be integrated with data-entry tools to enforce constraints at the time of metadata creation, but it is often used for assessment or quality control afterward. Despite this difference, validation code is useful to consider in any discussion of MAPs. Using it, we can record requirements for the same primary facets of a metadata model that we've discussed up to now—entities, properties, and values.

#### SHAPE EXPRESSIONS LANGUAGE

The Shape Expressions Language (ShEx) is focused specifically on RDF data and allows users to define conditions that an RDF graph should meet. As with other data-validation languages, it can be used with large quantities of data to quickly and accurately identify portions that don't conform to specified conditions.

Shape Expressions Compact Syntax (ShExC) provides a syntax for writing Shape Expressions schemas that is designed for human readability. ShExC is notated using terms that are relatively well-known from the RDF data model, widely used XML Schema data type terms, and intuitive labels for constraints. Because of this, users already familiar with RDF will have relatively little trouble reading and comprehending a ShExC schema (see figure 5.1), although writing one requires more detailed knowledge of the syntax.

Uptake and usage of this language may increase further given the announcement in May 2019 that Shape Expressions schemas would be enabled within the Wikidata platform, including the addition of a new entity type, the EntitySchema.<sup>8</sup> Schema numbers in Wikidata are prefixed with the letter E and encoded using ShExC.

#### SIMPLE DATA SERIALIZATIONS

The ShEx compact syntax provides one example of a relatively easy-to-read validation language that is

```

PREFIX ex: <http://example.org/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX schema: <https://schema.org/>
PREFIX dct: <http://purl.org/dc/terms/>
PREFIX dce: <http://purl.org/dc/elements/1.1/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX dcmitype: <http://purl.org/dc/dcmitype/>

ex:a_photo_shape {
  rdf:type [ schema:MediaObject ] {1} ;
  schema:contentUrl IRI + ;
  dct:title rdf:langString {1} ;
  dce:creator rdf:langString * ;
  dct:spatial IRI * ;
  dct:date xsd:date ? ;
  dct:type [ dcmitype::~ ] +
}

```

**Figure 5.1**

Requirements for RDF description of photographic resources expressed in ShExC syntax

machine-processable. While ShExC is used for the specific purpose of defining requirements for RDF data, it may be possible to utilize other general-purpose data formats for MAPs that offer the same combination of machine and human readability. The YAML data serialization language is one such format. This language has been designed for use in a wide variety of software implementations and makes use of an extremely simple syntax that preserves readability for humans.

Yet Another Metadata Application Profile (YAMA) markup language provides an interesting example of YAML syntax implementation. YAMA builds on the YAML syntax specification by creating a structure for recording the components of a MAP, specifying an extensible set of key/value pairs for defining entities, properties, and value constraints.<sup>9</sup> YAMA MAPs are well-suited to machine processing for output of validation code and other derivatives. Additionally, because they use the simple YAML syntax and natural-language key names for recording MAP content, they can be read by humans as well.

### Tools for Conversion

For implementers with sufficient technical know-how, it may be desirable to record MAP information only in a machine-readable presentation and use this version alone for reference. For many metadata creators and managers, however, this is not a realistic solution. Even when an encoded format is required to interact with data-entry forms, validate created metadata, or meet other needs, many of us will want a human-readable presentation as reference for ourselves and catalogers and metadata specialists who describe collections, and to make available to others who may wish to reuse our data. In these cases, the availability of

tools to generate machine-readable versions of MAPs from human-readable documents, or vice versa, will be important. Without such tools, we face the challenge of maintaining two distinct versions of the same MAP.

For MAPs that originate in machine-encoded form, various methods exist for generating human-readable documentation, including the use of templating engines written in JavaScript, Python, or other programming languages, and the use of XML stylesheets for MAPs encoded using XML or RDF/XML syntax. These can be used to output the information in an encoded MAP as HTML for viewing in a web browser or in PDF or another document format.

By supplementing or replacing machine-encoded notation with text meant for humans, the transformation process can provide MAP versions that are better suited for use by catalogers and other metadata specialists. Additional text can include natural-language descriptions of MAP components to replace encoding, which is often truncated for efficiency or embedded in syntax that is necessary for interpretation by software but hinders readability. The transformation process can also provide additional text meant for humans, such as instructions for recording values and properly formed examples.

Human-readable MAPs can provide a more detailed description of entities, properties, and value constraints than encoded versions, but these documents usually lack the structure needed for efficient processing using software tools, making the conversion of human- to machine-readable MAPs a more challenging task than moving in the other direction.

Providing human-readable MAPs in a more structured format may make the task of converting them to encoded MAPs easier. At the time of writing, the



DCMI Application Profiles Interest Group is developing a vocabulary for expressing human-readable MAPs in a standardized tabular form, and a Python software package is being developed in support of this work.<sup>10</sup> Two primary use cases for this software are validating the structure of MAPs written using the vocabulary and generating ShEx schemas based on these. These projects may allow a larger group of users to take advantage of data-validation capabilities by creating structured tabular MAPs and converting these to ShEx encoding.

## Notes

1. “DLF AIG Metadata Application Profile Clearinghouse Project,” Digital Library Federation Assessment Interest Group Metadata Working Group, <https://dlfmetadataassessment.github.io/MetadataSpecsClearinghouse/>.
2. “PCC Statistics,” Program for Cooperative Cataloging, Library of Congress, <https://www.loc.gov/aba/pcc/stats.html>.
3. “Wikidata:WikiProject PCC Wikidata Pilot/Participants,” Wikidata, last updated May 1, 2021, [https://www.wikidata.org/wiki/Wikidata:WikiProject\\_PCC\\_Wikidata\\_Pilot/Participants](https://www.wikidata.org/wiki/Wikidata:WikiProject_PCC_Wikidata_Pilot/Participants).

4. “Wikidata:WikiProject Books,” Wikidata, last updated October 6, 2020, [https://www.wikidata.org/wiki/Wikidata:WikiProject\\_Books](https://www.wikidata.org/wiki/Wikidata:WikiProject_Books).
5. “BIBFRAME Model, Vocabulary, Guidelines, Examples, Analyses,” Library of Congress, <https://www.loc.gov/bibframe/docs/index.html>.
6. “BIBFRAME Profile Editor,” Library of Congress, <http://bibframe.org/profile-edit>.
7. “Sinopia Vocabulary,” Linked Data Editor, Linked Data for Production 2 (LD4P2), <https://sinopia.io/vocabulary>.
8. “Shape Expressions arrive on Wikidata on May 28th,” Wikidata, [https://www.wikidata.org/wiki/Wikidata:Project\\_chat/Archive/2019/05#Shape\\_Expressions\\_arrive\\_on\\_Wikidata\\_on\\_May\\_28th](https://www.wikidata.org/wiki/Wikidata:Project_chat/Archive/2019/05#Shape_Expressions_arrive_on_Wikidata_on_May_28th).
9. Nishad Thalhath, Mitsuharu Nagamori, Tetsuo Sakaguchi, and Shigeo Sugimoto, “Yet Another Metadata Application Profile (YAMA): Authoring, Versioning and Publishing of Application Profiles,” *DC-2019—The Seoul, South Korea, Proceedings*, DCMI International Conference on Dublin Core and Metadata Applications, 114–25, <https://dcpapers.dublincore.org/pubs/article/view/4253>.
10. “dcmi/dctap,” Dublin Core Metadata Initiative, GitHub, <https://github.com/dcmi/dctap>; Tom Baker, “Tap2shex,” GitHub, 2021, <https://github.com/tombaker/tap2shex>.

# Examples

Theodore Gerontakos

The following examples show fragments of application profiles that demonstrate specific features of the AP process. The examples are described using data fields in accordance with a novel application profile drafted specifically to structure descriptions of AP fragments. That AP can be found below.

Metadata specialists often think in data chunks. Arguably, metadata professionals prefer reading data more than the expository text of the previous chapters. We'll enact that assertion here, offering our examples as data guided by our novelty AP for describing fragments of application profiles, which itself is more data. The features that we think are most interesting about the AP fragments are in the "Noteworthy Features of the Fragment" field as well as in the title/heading of each example.

In many circumstances, an "element set" or "schema" would have been improvised for describing AP fragments, consisting of a few unique properties. In order to create a true AP, we've selected components from already-existing sources and repurposed them for describing AP fragments. The AP is not professional-level chiefly because it is not machine-readable, but also because it is verbose and semi-structured data.

In the AP, eight properties are used to describe AP fragments; those eight properties are described here using the following seven properties, which may rightfully be called properties of properties of properties (demonstrating the notorious "meta" aspect of metadata work): (1) Property URI, (2) Label, (3) Sub-property Of, (4) Description, (5) Mandatory/Optional, (6) Range, (7) Note [optional property].

1. Property URI: <http://purl.org/dc/terms/title>
  - a. Label: <none>
  - b. Sub-property Of: <http://purl.org/dc/elements/1.1/title>
  - c. Description: Enter a title; start with a fragment number formatted as follows: Fragment #[x]:

<CR>. Next enter an attribute of the fragment, like "Professional-level", "Widely-adopted", "Platform-specific", etc. Next, for APs, enter "human-readable" or "machine-readable" followed by "AP." If the fragment is not an AP but part of the AP process, simply describe the part of the AP process represented; for example, "Domain model."

- d. Mandatory/Optional: Mandatory
  - e. Range: rdfs:Literal
  - f. Note: The title appears as a heading above the fragment, without a label.
2. Property URI: <http://purl.org/dc/terms/bibliographicCitation>
    - a. Label: Citation
      - i. Preferred citation format: Chicago Manual of Style
        1. Preferred edition: 17th edition
      - b. Sub-property Of: <http://purl.org/dc/terms/identifier>
      - c. Description: Citation to the full application profile, particularly if it is contained in a single document
      - d. Mandatory/Optional: Mandatory
      - e. Range: rdfs:Literal
    3. Property URI: CHOICE: EITHER (1) <http://purl.org/dc/terms/isPartOf> or (2) <http://purl.org/dc/elements/1.1/relation>
      - a. Label: (1) Part Of ; (2) Related Resource (Citation)
      - b. Sub-property Of: (1) <http://purl.org/dc/terms/relation>, (2) [Not applicable]
      - c. Description: Use (1) "Part Of" when citing a URL; use (2) "Related Resource" when providing a bibliographic description as rdfs:Literal. This field references the context or source in which the AP was found, like a website splash page, an article, a repository, etc.
      - d. Mandatory/Optional: Mandatory, one or the other property

- e. Range: (1) schema:URL for “Part Of”, (2) rdfs:Literal for “Related Resource”
4. Property URI: <<http://imaginaryURI.edu/fake/sourceOfFragmentNote>>
- Label: Source of Fragment Note
  - Sub-property Of: <<http://purl.org/dc/terms/description>>
  - Description: Free-text additional information about either the full application profile or the context/source
  - Mandatory/Optional: Optional
  - Range: rdfs:Literal
5. Property URI: <<http://purl.org/dc/elements/1.1/contributor>>
- Label: Contributor
  - Sub-property Of: [Not applicable]
  - Description: List contributor(s) to the AP; contributors can be creators of the AP, maintainers, even prominent users of the AP. Separate multiple contributors with a semicolon.
  - Mandatory/Optional: Optional
  - Range: rdfs:Literal
6. Property URI: <<http://purl.org/dc/terms/description>>
- Label: Noteworthy Features of the Fragment
  - Sub-property Of: <<http://purl.org/dc/elements/1.1/description>>
  - Description: Bulleted list containing free text describing characteristics of APs exemplified in the fragment.
  - Mandatory/Optional: Mandatory
  - Range: rdfs:Literal
7. Property URI: <<http://purl.org/dc/terms/relation>>
- Label: Related Resource
  - Sub-property Of: <<http://purl.org/dc/elements/1.1/relation>>
  - Description: Enter the URI of a related resource.
  - Mandatory/Optional: Optional
  - Range: schema:URL
8. Property URI: <<http://purl.org/dc/elements/1.1/description>>
- Label: Note on Related Resource
  - Sub-property Of: [Not applicable]
  - Description: Enter anything of note about the related resource cited in the “Related Resource” field.
  - Mandatory/Optional: Optional
  - Range: rdfs:Literal

## Fragment 1: Professional-Level Machine-Readable AP

*Citation:* Cataloging and Metadata Services, University of Washington Libraries, “WAU.profile.RDA.json,” *GitHub*, accessed April 7, 2021, <https://github.com/uwlib-cams/UWLibCatProfiles/blob/master/WAU.profile.RDA.json>.

*Part Of:* <https://sinopia.io>

*Source of Fragment Note:* No longer available as a full profile but as individual resource templates. Resource template for RDA Work (Monograph) available at <https://api.sinopia.io/resource/WAU:RT:RDA:Work:monograph>.

*Contributor:* University of Washington; LD4; LD4P2

*Noteworthy Features of the Fragment:*

- Based on the Library of Congress (LC) specification for BIBFRAME profiles available at <https://www.loc.gov/bibframe/docs/bibframe-profiles.htm>.
- Note that the LC specification is for BIBFRAME profiles but can be used for profiles other than BIBFRAME profiles; in this case, RDA.
- Authored in JSON.
- Includes full profile description, resource template description, and the property descriptions within the appropriate resource template.
- The fragment above includes local customizations, illustrating how the specification is extensible.

*Related Resource:* <https://uwlib-cams.github.io/web-views/rdaprofiles/WAU.profile.RDA.monograph.html>

*Note on Related Resource:* The related resource above is an RDA profile for monographs in a more human-readable form (derived from the machine-readable profile).

```

{
  "Profile": {
    "title": "WAU Profile RDA",
    "id": "WAU:profile:RDA",
    "author": "Ben Riesenbergr (ries07@uw.edu)",
    "date": "2020-01-22",
    "description": "RDA Work, Expression, Manifestation, Item, Agent, Nomen, Place, and
Timespan; BFLC Administrative Metadata and Status; MADS RDF Complex Subject and Sub-
ject; RDF List",
    "schema": "https://raw.githubusercontent.com/fakePath/uw-profile.json",
    "resourceTemplates": [
      {
        "resourceURI": "http://rdaregistry.info/Elements/c/C10001",
        "resourceLabel": "WAU RT RDA Work",
        "id": "WAU:RT:RDA:Work",
        "date": "2020-01-22",
        "author": "Ben Riesenbergr (ries07@uw.edu)",
        "schema": "https://raw.githubusercontent.com/fakePath/uw-resource-template.json",
        "propertyTemplates": [
          {
            "propertyLabel": "has preferred title of work (*) (RDA 6.2.2)",
            "propertyURI": "http://rdaregistry.info/Elements/w/P10223",
            "mandatory": "true",
            "repeatable": "false",
            "type": "literal",
            "uwFormOrder": 0.004,
            "usedInProfile": [ "etd", "map", "eMap", "eBook" ],
            "remark": "http://access.rdatoolkit.org/6.2.2.html"
          },
          {
            "propertyLabel": "has creator agent of work (RDA 19.2)",
            "propertyURI": "http://rdaregistry.info/Elements/w/P10065",
            "mandatory": "false",
            "repeatable": "true",
            "type": "lookup",
            "valueConstraint": {
              "useValuesFrom": ["urn:ld4p:qa:names"]
            },
            "uwFormOrder": 161,
            "usedInProfile": [ "monograph", "dvdVideo", "eMap", "eBook" ],
            "remark": "http://access.rdatoolkit.org/19.2.html"
          }
        ]
      }
    ]
  }
}

```

**Figure 6.1**  
University of Washington's Sinopia profile for RDA (fragment)

#### 4.1.A class = dpla:SourceResource

This class is a subclass of "edm:ProvidedCHO," which comprises the described resources (in EDM called "cultural heritage objects") about which DPLA collects descriptions. It is here that attributes of the described resources are located, not the digital representations of them.

Label	Property	Sub-property of	Range	Usage	Data Type	Vocab/Syntax Schema	Obligation
Alternate Title	dcterms:alternative .sourceResource. alternative	dc:title		Any alternative title of the described resource including abbreviations and translations.	Literal		0 – n
Collection*	dcterms:isPartOf, .sourceResource. collection	dc:relation	dc:type: Collection	Collection or aggregation of which described resource is a part.	Reference		0 – n
Contributor	dcterms: contributor .sourceResource. contributor		edm: Agent	Entity responsible for making contributions to described resource.	Reference		0 – n
<b>Creator*</b>	dcterms:creator .sourceResource. creator		edm: Agent	Entity primarily responsible for making described resource.	Reference		0 – n

**Figure 6.2**  
The Digital Public Library of America (DPLA) MAP, version 5.0 (fragment)

### Fragment 2: Professional-Level Human-Readable AP

*Citation:* DPLA MAP Working Group, *Metadata Application Profile, version 5.0*, Digital Public Library of America, 2017, <http://dp.la/info/map>.

*Part Of:* <https://pro.dp.la/hubs/metadata-application-profile>

*Source of Fragment Note:* The AP is still available only as a human-readable PDF file.

*Contributor:* Digital Public Library of America

*Noteworthy Features of the Fragment:*

- Fully modernized AP except it is not machine-readable.
- The properties in Figure 6.2 are used only with instances of the dpla:SourceResource class;

this means property domains are implicitly enumerated.

- Several properties also enumerate their range.
- Data type, or “node type”, of values is also specified.
- Note how human-readable documents can describe properties using formatting (boldface text to signify that the property appears in the DPLA portal) and symbols (an asterisk to signify that the property is recommended).
- The overall model is based largely on the Europeana Data Model.

*Related Resource:* <https://pro.europeana.eu/page/edm-documentation>

*Note on Related Resource:* The related resource cited above is the “Europeana Data Model” web page, a collection of documents; note especially the “EDM definition.”

RDA Instructions & Elements	RDA No.	Notes	MARC Encoding
Identifying Works & Expressions		The authorized access point for principal creator (if any) is required for use in conjunction with the work and expression attributes listed in this section (see also RDA 19.2 below). If a formal authorized access point for a work or expression is also included in the BSR, its form should be established following NACO policies.	
Preferred title for work -- Musical work -- Legal work -- Religious work -- Official communication	6.2.2 6.14.2 6.19.2 6.23.2 6.26.2	Record as part of an authorized access point if the preferred title for work differs from the title proper (245 \$a \$n \$p) or if additional differentiating elements are needed.	130, 240, 7XX
Form of work	6.3	Record if needed to differentiate.  For assigning genre/form terms in 6XX fields, see "Subject and genre/form access" in the "Required Non-RDA and MARC Data" sections below.	130, 240, 380, 7XX
Date of work -- Legal work -- Treaty	6.4 6.20 6.20.3	Record if needed to differentiate. Always record the date of a treaty.	046, 130, 240, 7XX
Place of origin of work	6.5	Record if needed to differentiate.	130, 240, 7XX
Other distinguishing characteristic of work -- Legal work	6.6 6.21	Record if needed to differentiate.	130, 240, 381, 7XX

**Figure 6.3**  
BIBCO Standard Record RDA MAP: RDA Core and PCC Core Elements: Identifying Works and Expressions (fragment)

### Fragment 3: Widely Adopted Human-Readable AP for Library Data

*Citation:* Program for Cooperative Cataloging, *BIBCO Standard Record (BSR) RDA Metadata Application Profile*, Program for Cooperative Cataloging, 2020, <https://www.loc.gov/aba/pcc/bibco/documents/PCC-RDA-BSR.pdf>.

*Part Of:* <https://www.loc.gov/aba/pcc/bibco/bsr-maps.html>

*Source of Fragment Note:* The citation above references the 2020 revision; note that, in the web page containing the link to the BSR, historical versions are available.

*Contributor:* PCC; BIBCO

*Noteworthy Features of the Fragment:*

- Recommendation of required minimal description in a shared cataloging environment.
- Not a full-featured AP; highly pragmatic AP.
- Makes use of RDA entities and relationships.
- Bridge between MARC-based past and linked-data future.

*Related Resource:* <https://www.loc.gov/aba/pcc/conser/documents/CONSER-RDA-CSR.pdf>

*Note on Related Resource:* The related resource above is PCC's CONSER Standard Record; a link to the document is available on PCC's web page at <https://www.loc.gov/aba/pcc/conser/index.html>.

## Fragment 4: Platform-Specific AP

*Citation:* DSpace, version 5.6, DuraSpace, 2016.

*Part Of:* <https://digital.lib.washington.edu/researchworks/>

*Source of Fragment Note:* This schema is active in University of Washington, “Metadata Registry,” *ResearchWorks Archive*, accessed April 7, 2021.

*Contributor:* DSpace; DuraSpace; Lyrasis

*Noteworthy Features of the Fragment:*

- This is the “dc” AP that came with DSpace 5.6 out of the box.
- Note that the AP is extensible; we can add custom local fields.
- The AP is extremely minimal: a list of properties with optional “scope notes.”
- The properties are associated with an IRI.
- The AP is an extension of and variation on Dublin Core properties.

*Related Resource:* <https://duraspace.org/dspace/>

*Note on Related Resource:* The related resource cited above is the website for the current version of DSpace (version 7).

## Fragment 5: Domain Model

*Citation:* Dublin Core Metadata Initiative, “DC-Ed-Model—2009-12-09.jpg,” *GitHub*, accessed April 8, 2021, [https://github.com/dcml/repository/blob/master/wikis\\_pre2016/education/educationwiki/Model\\_files/DC-Ed-Model--2009-12-09.jpg](https://github.com/dcml/repository/blob/master/wikis_pre2016/education/educationwiki/Model_files/DC-Ed-Model--2009-12-09.jpg).

*Related Resource (Citation):* Phil Barker and Lorna Campbell, “Metadata for Learning Materials: An Overview of Existing Standards and Current Developments,” *Technology, Instruction, Cognition and Learning* 7 (2010): 225–43, [https://www.researchgate.net/publication/228802531\\_Metadata\\_for\\_learning\\_materials\\_An\\_overview\\_of\\_existing\\_standards\\_and\\_current\\_developments](https://www.researchgate.net/publication/228802531_Metadata_for_learning_materials_An_overview_of_existing_standards_and_current_developments).

Metadata Schema: "dc"

This is the metadata schema for "http://dublincore.org/documents/dcmi-terms/". You may add new or update existing metadata fields to this schema. Fields may also be selected for deletion or be moved to another schema.

Add new metadata field

Field Name: dc .

Scope Note:

Additional notes about this metadata field.

Add new metadata field

Schema metadata fields

ID	Field	Scope Note
<input type="checkbox"/> 2	dc.contributor.advisor	Use primarily for thesis advisor.
<input type="checkbox"/> 3	dc.contributor.author	
<input type="checkbox"/> 4	dc.contributor.editor	
<input type="checkbox"/> 5	dc.contributor.illustrator	
<input type="checkbox"/> 6	dc.contributor.other	
<input type="checkbox"/> 1	dc.contributor	A person, organization, or service responsible for the content of the resource. Catch-all for unspecified contributors.
<input type="checkbox"/> 7	dc.coverage.spatial	Spatial characteristics of content.

**Figure 6.4**  
DSpace “dc” Metadata Schema (fragment)

*Source of Fragment Note:* There does not appear to be a completed domain model, although the model presented can be considered a complete domain model; the model presented here appeared in the article cited above.

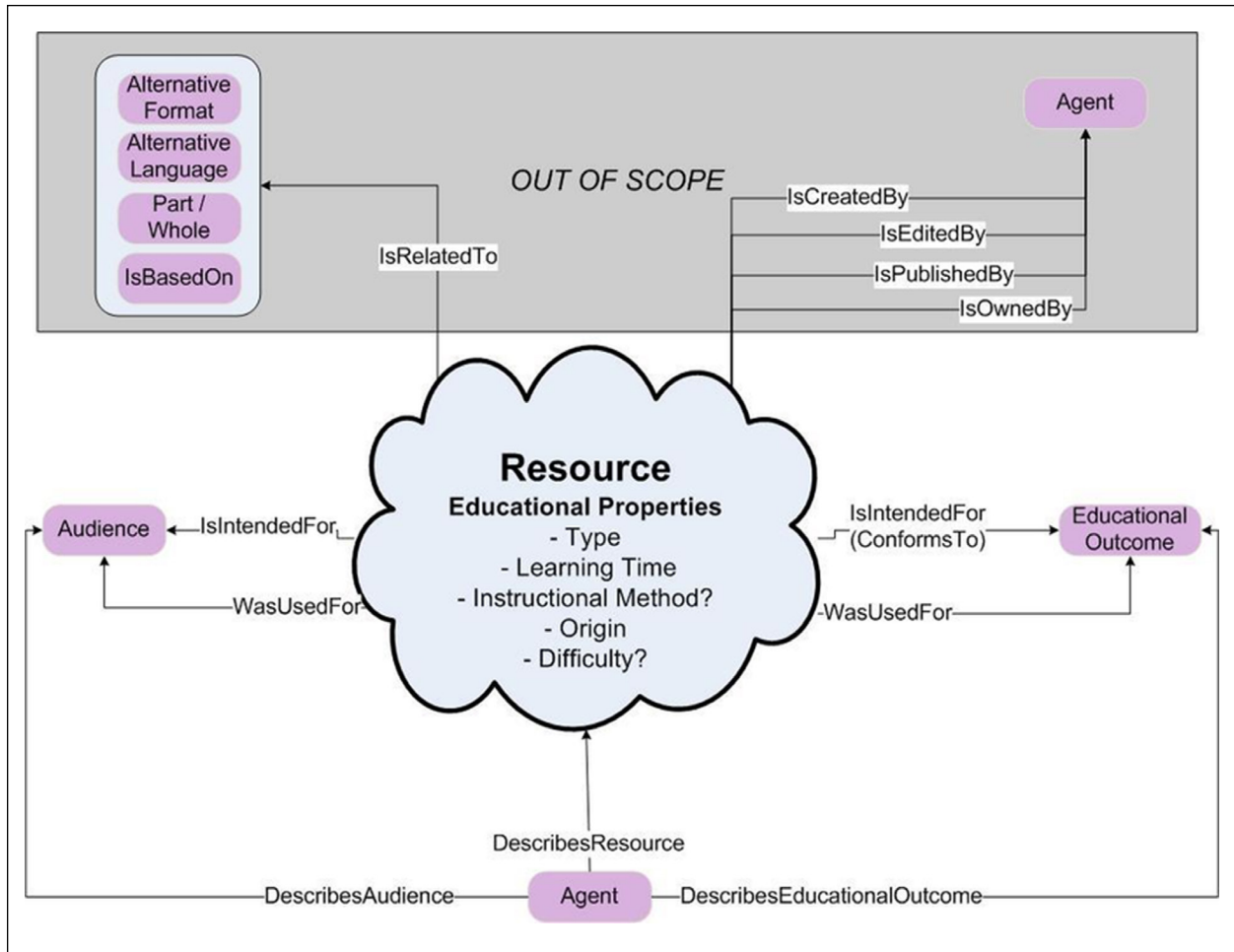
*Contributor:* Dublin Core Metadata Initiative (DCMI)

*Noteworthy Features of the Fragment:*

- This was not considered a complete domain model when proposed around 2010.
- One main entity “Resource” seems required; each instance would require an IRI.
- Three additional entities may require IRIs.
- Some out-of-scope entities are explicit.
- Relationships between entities are explicit.

*Related Resource:* <https://dublincore.org/groups/education>

*Note on Related Resource:* The related resource above is the archived page for the DCMI Education Community (no longer active).



**Figure 6.5**  
Draft domain model for the DCMI DC-Education profile



# Library Technology

## R E P O R T S

Upcoming Issues	
October 57:7	<b>Library IT Management in Times of Crisis</b> by Jason Bengtson
November/ December 57:8	<b>Using the Toward Gigabits Libraries Toolkit</b> by Carson Block
January 58:1	<b>Introduction to Library Engagement Platforms</b> by David Lee King

### Subscribe

[alatechsource.org/subscribe](http://alatechsource.org/subscribe)

### Purchase single copies in the ALA Store

[alastore.ala.org](http://alastore.ala.org)



ALATechSource

[alatechsource.org](http://alatechsource.org)

ALA TechSource, a unit of the publishing department of the American Library Association