ANSI/NISO Z39.98-2012

Authoring and Interchange Framework for Adaptive XML Publishing Specification

Abstract: This specification defines a framework in which to develop XML markup languages to represent different kinds of information resources (books, periodicals, etc.), with the intent of producing documents suitable for transformation into different universally accessible formats. It uses a modular, extensible architecture to permit the creation of any number of document models, each custom-tailored for a particular kind of information resource.

An American National Standard
Developed by the
National Information Standards Organization

Approved: July 9, 2012
by the
American National Standards Institute

Published by the National Information Standards Organization
Baltimore, MD, USA

About NISO Standards

NISO standards are developed by the Working Groups of the National Information Standards Organization (NISO) with oversight from a Topic Committee. The development process is a strenuous one that includes a rigorous peer review of proposed standards open to each NISO Voting Member and any other interested party. Final approval of the standard involves verification by the American National Standards Institute that its requirements for due process, consensus, and other approval criteria have been met by NISO. Once verified and approved, NISO Standards also become American National Standards.

This standard may be revised or withdrawn at any time. For current information on the status of this standard contact the NISO office or visit the NISO website at: www.niso.org.

Published by:
National Information Standards Organization (NISO)
One North Charles Street, Suite 1905
Baltimore, MD 21201
www.niso.org

ISBN: 978-1-937522-06-3 (HTML)

ISBN: 978-1-937522-07-0 (PDF)

Table of Contents

List of Examples

Foreword

(This foreword is not part of the *Authoring and Interchange Framework for Adaptive XML Publishing Specification*, ANSI/NISO Z39.98-2012. It is included for information only.)

About This Standard

The Z39.98 *Authoring and Interchange Framework for Adaptive XML Publishing Specification* (Z39.98-AI) defines a framework in which to develop XML markup languages to represent different kinds of information resources (books, periodicals, etc.), with the intent of producing documents suitable for transformation into different universally accessible formats. It uses a modular, extensible architecture to permit the creation of any number of document models, each custom-tailored for a particular kind of information resource.

This approach to text production differs significantly from the one taken in ANSI/NISO Z39.86-2005 (the DTBook grammar), which attempted to provide a single markup model in which all formats could be present (i.e., a single universal rendering format). Although the Z39.86 approach has potential merits for reader consumption, it could not provide the richness needed by producers in many cases to render high-quality individual outputs. Producing print braille compliant to regional codes, as one example, was complicated by markup that was often more generally useful for refreshable braille display. Accessible production by needs requires the ability to repurpose content in a variety of forms for readers of different abilities, and as efficiently as possible, but this need that DTBook also hoped to address was not being fully realized.

The Z39.86 text model was further complicated in that it could not be easily redefined for specific use cases. All content had to be structured exactly the same way, regardless of the form it took in its source. By focusing on accessible output requirements, it was also not widely useful as a production format for mainstream publishing requirements, limiting the ability to obtain content from source producers.

This standard escapes the trap of defining markup models and instead focuses on a general, extensible and highly-adaptable framework in which content models can be defined. It prescribes the rules and requirements for predictable and rapid development of new content models without defining specifically defining the grammars. The development of single source master documents that can be easily exchanged between organizations is the ultimate goal, but without imposing limits on the markup expressivity needs of any individual producer.

The richness of markup that can be produced using this model also places this standard back in the mainstream. Its focus on fully representing the structure and meaning of the documents being described makes it a candidate for use in any environment in which a parallel publishing model is currently used or envisioned. The outputs that can be generated from documents that conform to Z39.98-AI profiles are not limited to accessible formats.

Z39.98-AI was originally intended to be a revision to and replacement for ANSI/NISO Z39.86, *Specifications for the Digital Talking Book.* After consideration of feedback from the draft for

trial use of the proposed revision, the Working Group recommended that the revision be given a new standard designation number and that the existing Z39.86 standard be reaffirmed. Trial users had indicated that the changes were so significant as to warrant this being a new standard. Additionally, content creators, software developers, and e-reader device manufacturers wanted to continue using the existing standard for the near future while they developed transition plans to the new standard. The NISO Content and Collection Management Topic Committee approved the Working Group's recommendation and this standard was assigned the new designation of Z39.98. Subsequently, ANSI/NISO Z39.86 was reaffirmed for another five years.

Trademarks, Services Marks

Wherever used in this standard, all terms that are trademarks or service marks are and remain the property of their respective owners.

NISO Voting Pool

At the time this standard was approved, the following were members of the NISO Voting Pool:

American Library Association (ALA)
Nancy Kraft

American Psychological Association
Linda Beebe, Janice Fleming

American Society for Indexing
JudithGibbs

American Society for Information Science & Technology (ASIS&T)
Mark Needleman

Association of Research Libraries (ARL)
Julia Blixrud, Charles Lowry

College Center for Library Automation (CCLA)
David Brightbill, Lucy Harrison

DAISY Consortium
George Kerscher, Markus Gylling

Inera Inc.
Bruce Rosenblum

ITHAKA/JSTOR/Portico
Bruce Heterick, Amy Kirchhoff

Library of Congress
Sally McCallum, John Zagas

Lyrasis
Robin Dale, Tim Daniels, Peter Murray

Music Library Association
Mark McKnight. David Sommerfield

National Archives and Records Administration (NARA)
Laura McCarthy, Marilyn Redman

National Library of Medicine (NLM)
Barbara Rapp, Jacque-Lynne Schulman

National Security Agency
Kate Dolan, Kathleen Rattell

Polaris Library Systems
Eric Graham, Paul Huf

Recording Industry Association of America (RIAA)
David Hughes, Paul Jessop

NISO Content and Collection Management Topic Committee

At the time this standard was approved, the following individuals served on the NISO Content and Collection Management Topic Committee that had oversight for the development of this standard:

Julia Blixrud
Association of Research Libraries (ARL)

Eva Bolkovac
Yale University Library

Lettie Conrad
SAGE Publications

Diane Hillmann
Syracuse University

Marjorie Hlava
Access Innovations, Inc.

Rebecca Kennison
Columbia University

Betty Landesman
NIH Library

Rice Majors
University of Colorado at Boulder

Dorothea Salo
University of Wisconsin, Madison

Ken Wells
Innovative Interfaces, Inc.

NISO Z39.98-AI Working Group

The following are the members of the Z39.98-AI Working Group responsible for the development of this standard:

Markus Gylling, Lead
DAISY Consortium

Josh Altherr
gh, LLC

Ole Holst Andersen
DBB

Marisa DeMeglio
DAISY Consortium

Christian Egli
Swiss Library for the Blind, Visually Impaired and Print Disabled

Matt Garrish
CNIB

Boris Goldowsky
CAST, Inc.

Leona Holloway
Vision Australia

Kenny Johar
Vision Australia

Dennis Leas
gh, LLC

Sam Ogami
California State University

Stephen Phippen
RNIB

James Pritchett
Learning Ally

Kathryn Randall
Vision Australia

Per Sennels
Huseby kompetansesenter

The metadata portion of this standard was created by a working group with the following members:

Matt Garrish, Lead
CNIB

Bob Axtell
Library of Congress

Wendy Taylor
RNIB

Christian Wallin
DBB

Marcus Westlind
TPB

Richard Wilson
CNIB

1 Purpose and Scope

1.1 Audience

This specification details the nature of Z39.98 Authoring and Interchange Format (Z39.98-AI) profiles and how they are created. It is intended primarily for agencies interested in creating conformant profiles for new documents types and for processing agent developers.

This specification is not a guide to marking up Z39.98-AI documents and should not be referenced as such. Informative resources that describe Z39.98-AI document production are available independently of this specification.

Although this specification contains introductory sections where appropriate and deals with general document concepts in places, it is expected that all persons reading this specification will have a strong background in XML and its related technologies—in particular schema languages and their composition—in order to properly implement new profiles.

1.2 Design goals

The Z39.98-AI Framework has been built with the following primary design goals in mind:

1. *Adaptability*. The Framework is designed to be flexible and customizable across a wide variety of production environments. Producers are not locked into a pre-defined schema, but can use the built-in mechanisms for customization and extension to fit the Framework to their specific requirements: profiles adapt to fit the information resources they describe instead of the other way around, localizations are easily implemented, and deployment in production environments spanning the gamut of markup complexity—from complex/rich to simple/reduced—is easily accommodated.
2. *Modularity*. Framework profiles are created using discrete schema modules, building on and incorporating existing profiles and standards where possible. The schema modules created for a profile, in turn, can be reused when building other profiles, progressively decreasing the work required to create each new profile. The modular aspect of the framework also simplifies the development of processing tools by similarly reducing the number of custom components needed to accommodate new profiles.
3. *Self-describing*. The Framework is designed to move beyond the often contentious issue of XML tag names alone defining the semantics of the structures they represent. Finer control over the intent and meaning of markup is now offered through the layering of RDF metadata. The flexibility to attach multiple bibliographic record types to a document to meet the various needs it will serve is also now provided.
4. *Data repurposing*. The primary goal of the specification is to facilitate the parallel publishing of documents through its open modular framework. Z39.98-AI profiles provide the essential structures that compose documents in an unambiguous and format-agnostic way, and are like a master blueprint to the information resources they describe. As a result, Z39.98-AI documents can be manipulated through automated transformation

chains—such as specified by the XProc standard [XProc]—to create output formats (e.g., print, EPUB, etc.) and to create the inputs for alternate publishing processes (e.g., braille).

1.3 Overview

Creating profiles that conform to this specification requires understanding how the various concepts and technologies outlined in this document are bound together to define information resources. Although some readers will be sufficiently comfortable with XML technologies to jump straight into the specification, this overview provides a quick reference guide to how profiles are constructed and where the requirements are defined for both novice and experienced developers alike.

At the core of the Z39.98-AI specification is the Abstract Document Model, which introduces the framework common across all profile implementations, and the high-level rules to which profile creators must adhere in order to ensure consistency and predictability between grammars. The Abstract Document Model is like a map to building profiles and its concepts are fully introduced in 4, *Abstract Document Model*.

Understanding the Abstract Document Model is essential to understanding the rest of the specification, as the process of creating profiles involves formalizing rules to enforce the abstract concepts, as illustrated in Figure 1.

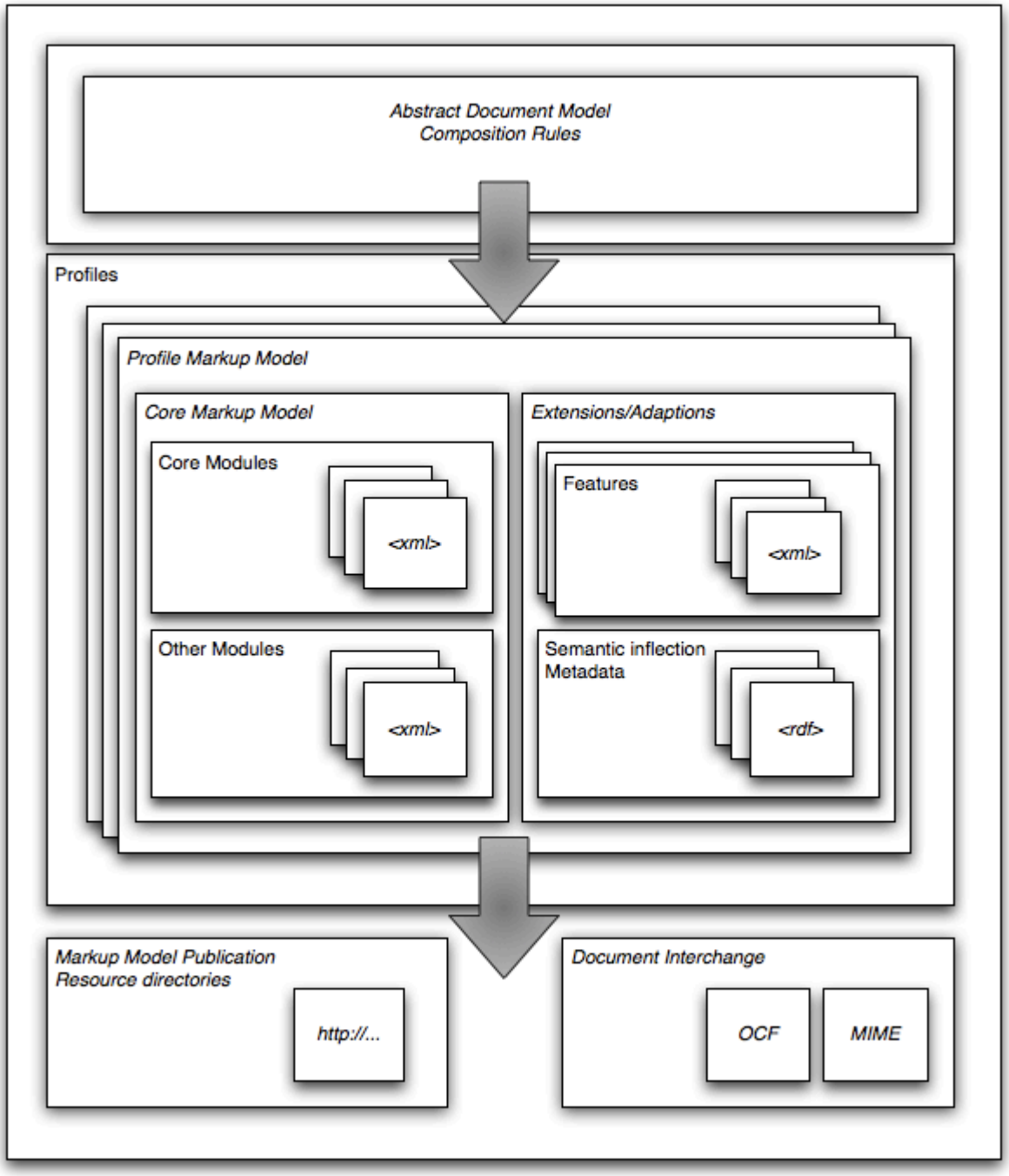

Figure 1: Overview of the Z39.98-AI profile creation process

Profiles are the practical product of the Z39.98-AI specification and take the form of markup models defining the structure of information resources. The rules and requirements for creating them are detailed in 6, *Profiles*.

Profiles are built on a modular model, allowing component definitions to be reused across Z39.98-AI profiles and to be included from other industry-standard grammars. The following sections of the specification introduce the various parts in their construction:

- 5, *Modules* – Modules are sets of elements and attributes that are semantically and/or structurally linked through the traits they exhibit. Modules are activated when creating profiles and their components become the building blocks for composing the new grammar.
- 5.6, *Core modules* – The core modules are a set of modules developed by the Z39.98-AI Working Group to encourage reuse of components across profiles.
- 7, *Features* – Like a cross between profiles in miniature and highly-specialized modules, features provide comprehensive markup to represent very particular kinds of structures (e.g. MathML, Ruby, etc.). Features help ensure the consistency of specialized markup between Z39.98-AI documents and ensure that Z39.98-AI profiles are properly aligned with industry standards.

RDF is the primary means provided by the Framework for the expression of metadata about documents and for the semantic inflection of meaning on elements they contain. Although profile creators may use other methods to annotate their data, the use of RDF vocabularies is encouraged because of their tight binding to the specification. Information on how to apply the available attributes and vocabularies can be found in 11, *RDF vocabularies*.

A completed profile will typically consist of a variety of different resources, such as schema files, RDF vocabularies, additional prose restrictions as well as usage and other documentation. The publishing of a profile is the act of assigning it an identity URI (see 6.3.1, *Profile identity URI*), at which location a resource directory document is available that enumerates these resources and provides additional information on how to obtain them. Complete information on how to create resource directories and documents is available in 10, *Resource directories*.

This specification also includes a catalog of profiles that have been made publicly available for use in creating documents (available in Appendix A, *Profile, feature, and vocabulary catalogs*). Producers are not required to use these profiles, but they have been designed for the widest use possible to encourage their adoption. The profiles represent fully-compliant implementations of this specification for developers or individuals looking for practical representations of the specification.

Although this specification is not specifically intended for document creators, it does contain information about the general nature of document creation, including how to specify the profile a document conforms to, any features in use, as well as required metadata that must be included to identify the document. This information can be found in 8, *Documents*.

This specification also includes a packaging format based on the Open Container Format [OCF], which may be used to bundle the XML, image, and other local resources that compose a Z39.98-AI document set. It also specifies a MIME type [RFC2046] for the container format to facilitate the interchange of these files. This information can be found in 9, *Container* and in Appendix C, *Media type registration*.

In addition to gaining a full understanding of all the topics outlined above, processing agent developers must also ensure that their applications meet the conformance requirements detailed in 12.2, *Processing agent conformance definition*.

1.4 Relationship to other specifications

1.4.1 ANSI/NISO Z39.86-2005

ANSI/NISO Z39.98-2012 represents an entirely new technical direction that supersedes Section 4, *Content Format for Text* of ANSI/NISO Z39.86-2005, *Specifications for the Digital Talking Book* [Z39.86-2005]. As such, text content files compliant under ANSI/NISO Z39.86-2005 will not be compliant under ANSI/NISO Z39.98-2012 and vice versa.

1.4.2 XML

The document types defined by ANSI/NISO Z39.98-2012 inherit all constraints defined by Extensible Markup Language (XML) 1.0 (Fifth Edition) [XML].

1.4.3 Unicode

This specification inherits its relation to Unicode from [XML].

1.5 Conformance statements

The keywords “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in [RFC2119].

The normative and informative appendices to this specification are identified by the labels “(normative)” and “(informative)” in their titles, respectively.

All examples in this specification are informative.

2 Normative References

The following standards contain provisions which, through reference in this text, constitute provisions of this standard. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this standard are encouraged to investigate the possibility of applying the most recent editions of the standards indicated below.

[DCMES]
ANSI/NISO Z39.85-2007, The Dublin Core Metadata Element Set.
NISO (National Information Standards Organization). 11 October 2010.
http://www.niso.org/standards/z39-85-2007/

[ISOSchematron]
ISO/IEC 19757-3: *Rule-based validation — Schematron.*
International Organization for Standardization/International Electrotechnical Commission. 01 June 2006.
http://standards.iso.org/ittf/PubliclyAvailableStandards/c040833_ISO_IEC_19757-3_2006(E).zip

[ITS]
Internationalization Tag Set (ITS) Version 1.0.
W3C (World Wide Web Consortium). 3 April 2007.
http://www.w3.org/TR/2007/REC-its-20070403/

[NVDL]
ISO/IEC 19757-4: *NVDL (Namespace-based Validation Dispatching Language).*
International Organization for Standardization/International Electrotechnical Commission. 01 June 2006.
http://standards.iso.org/ittf/PubliclyAvailableStandards/c038615_ISO_IEC_19757-4_2006(E).zip

[OCF]
EPUB Open Container Format (OCF) 2.0.1.
IDPF (International Digital Publishing Forum). 4 September 2010.
http://idpf.org/epub/20/spec/OCF_2.0.1_draft.doc

[RDFa]
W3C Working Draft: RDFa Core 1.1.
W3C (World Wide Web Consortium). 15 December 2012.
http://www.w3.org/TR/2011/WD-rdfa-core-20111215/

[XHTML+RDFa]
W3C Working Draft: XHTML+RDFa 1.1: Support for RDFa via XHTML Modularization.
W3C (World Wide Web Consortium). 15 December 2012.
http://www.w3.org/TR/2011/WD-xhtml-rdfa-20111215/

[RelaxNG]
ISO/IEC 19757-2: *Regular-grammar-based validation — RELAX NG*.
International Organization for Standardization/International Electrotechnical Commission. 10 December 2008.
http://standards.iso.org/ittf/PubliclyAvailableStandards/c052348_ISO_IEC_19757-2_2008(E).zip

[RFC1738]
RFC 1738: *Uniform Resource Locators (URL)*.
IETF (Internet Engineering Task Force). December 1994.
http://www.ietf.org/rfc/rfc1738.txt

[RFC2046]
RFC 2046: *Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types*.
IETF (Internet Engineering Task Force). November 1996.
http://www.ietf.org/rfc/rfc2046.txt

[RFC2119]
RFC 2119: *Key words for use in RFCs to Indicate Requirement Levels*.
IETF (Internet Engineering Task Force). March 1997.
http://www.ietf.org/rfc/rfc2119.txt

[RFC3023]
RFC 3023: *XML Media Types*.
IETF (Internet Engineering Task Force). January 2001.
http://www.ietf.org/rfc/rfc3023.txt

[RFC3986]
RFC3986: *Uniform Resource Identifier (URI): Generic Syntax*.
IETF (Internet Engineering Task Force). January 2005.
http://www.ietf.org/rfc/rfc3986.txt

[XInclude]
XML Inclusions (XInclude) Version 1.0 (Second Edition).
W3C (World Wide Web Consortium). 15 November 2006.
http://www.w3.org/TR/xinclude/

[XML]
Extensible Markup Language (XML) 1.0 (Fifth Edition).
W3C (World Wide Web Consortium). 26 November 2008.
http://www.w3.org/TR/2008/REC-xml-20081126/

[XMLBase]
XML Base (Second Edition).
W3C (World Wide Web Consortium). 28 January 2009.
http://www.w3.org/TR/2009/REC-xmlbase-20090128/

[XMLID]
xml:id Version 1.0.
W3C (World Wide Web Consortium). 9 September 2005.
http://www.w3.org/TR/2005/REC-xml-id-20050909/

[XMLInfoset]
XML Information Set (Second Edition).
W3C (World Wide Web Consortium). 4 February 2004.
http://www.w3.org/TR/2004/REC-xml-infoset-20040204/

[XMLNAMES]
Namespaces in XML 1.0 (Third Edition).
W3C (World Wide Web Consortium). 8 December 2009.
http://www.w3.org/TR/2009/REC-xml-names-20091208/

[XML Schema Part1]
XML Schema Part 1: Structures Second Edition.
W3C (World Wide Web Consortium). 28 October 2004.
http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/

[XML Schema Part2]
XML Schema Part 2: Datatypes Second Edition.
W3C (World Wide Web Consortium). 28 October 2004.
http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/

[XMLSTYLE]
Associating Style Sheets with XML documents 1.0 (Second Edition).
W3C (World Wide Web Consortium). 28 October 2010.
http://www.w3.org/TR/2010/REC-xml-stylesheet-20101028/

3 Terms and Definitions

class
: An abstract means of referring to a set of element and/or attribute declarations associated with a layer or set in the Abstract Document Model. Classes correspond to name classes in [RelaxNG] and named groups in [XML Schema Part1].

NOTE: Classes are dynamically populated depending on which modules and components are activated in a profile. Classes do not constitute content model templates, but are collections of declarations that can be used to create concrete content models.

component
: An individual element, attribute, datatype, value, or pattern as defined in a module or feature.

customization
: An element or attribute whose usage context, content model, and/or attribute model have been altered from their default state during activation.

feature
: A partial markup model designed to represent a limited, highly-specialized set of content structures. Mathematical equations, chemistry formulas, and musical notations are examples of the kinds of content structures that might be addressed by a Z39.98-AI feature. Features share the same general structure as profiles, but are more specialized and of narrower scope and are intended to be used as discrete components within profiles.

implicit value
: The value a processing agent must assume for an attribute when the attribute is not present. Implicit value declarations may apply to all elements that can contain the attribute or only to elements in particular contexts.

NOTE: Each component's definition defines the implicit values a processing agent must apply.

markup model
: The vocabulary and grammar of a profile or feature, as defined by a schema. The markup model encapsulates all the permutations to which valid documents and fragments must conform.

module
: An abstract unit within a markup model expressed as a schema fragment, used to consolidate markup declarations to increase the flexibility, modifiability, reuse, and understanding of specific logical or semantic structures.

module activation
: The act of including a module in a profile, thereby including all or some of its components in the markup model. Activation may include making alterations to the traits of the module components, as well as excluding components entirely.

module/component definition
: A human readable set of statements describing the default state and nature of a module and/or its components prior to activation, using a mixture of prose, RelaxNG Compact syntax [RelaxNG], and other means.

NOTE: The module and component definitions must not be relied on to provide information about the practical implementation of that module or component as employed in any Z39.98-AI profile.

processing agent
: An application that processes a Z39.98-AI document. Examples include, but are not limited to, authoring tools (XML editors, XML-enabled word processors), transformation pipelines, business transfer chains, and conformance validators.

NOTE: The domain of processing agents defined by this specification only encompasses the lifecycle of a Z39.98-AI document's creation, transformation, and provision. End user tools that render the content of a document, whether the Z39.98-AI document source or the outputs generated from it, do not fall under the scope or conformance requirements outlined in this specification.

NOTE: This definition does not include XML Processor as defined in [XML].

profile
: A markup model, associated RDF vocabulary, and normative prose describing their usage, designed as an integrated approach to represent information resources of a particular type in XML.

RDF vocabulary
: A vocabulary of terms that provides a mechanism to annotate elements or element content with machine-extractable semantic information about their nature or purpose.

resource directory
: A package of information regarding a profile or feature, including normative schemas, informative schemas, RDF vocabularies, documentation, style sheets, or other associated resources. Resource directories are expressed in XHTML+RDFa [XHTML+RDFa].

specialization
: An element or attribute that inherits some or all of its traits from another element, but that serves a more specific semantic purpose.

NOTE: Elements contributed to a layer in the Abstract Document Model typically inherit a set of basic traits from that layer's default member, and are thus specializations of the default member. Specializations may be derived from any member of a layer, in which case the default member's traits are only indirectly inherited.

NOTE: Unlike variants, specializations do not share the same QName with the element they inherit their traits from.

variant
: An element that derives its semantic definition from another element but that is intended for use in a different layer of the Abstract Document Model than the element it is derived from.

NOTE: A variant shares the same QName as the element it is derived from.

Z39.98-AI document
: An XML document that conforms to a Z39.98-AI profile.

Z39.98-AI document set
: A Z39.98-AI document together with any local and remote resources referenced by it.

NOTE: Local resources are defined as those that are referenced in a Z39.98-AI document by a file URI scheme [RFC1738] and must accompany the document in order to create a valid representation of it, while remote resources do not directly accompany the document but can be obtained according to their URI schemes.

4 Abstract Document Model

4.1 Introduction

The Abstract Document Model underpins the Z39.98-AI specification, providing the base structure through which all document profiles are defined. It is both a concept for defining the structure of documents and a model for creating valid Z39.98-AI profiles: the central concept of

the model being that documents can be decomposed to a common foundation and that functionality can be built up in distinct layers.

Four such layers of increasing structural markup granularity are defined to cover the major components of a document: a general section layer for the hierarchical structure, a block layer for container elements, a layer for phrase constructs, and finally the layer containing the text data of the document. The Abstract Document Model does not just define element structure, however; it additionally defines two complementary collections of attributes that flow through the element layers (the global and metadata attribute collections). It also exposes all of these layers and collections for customization and extension, which is where its flexibility for representing new document types lies.

Figure 2 provides a visual perspective of how the layers work in a bounding box-like fashion.

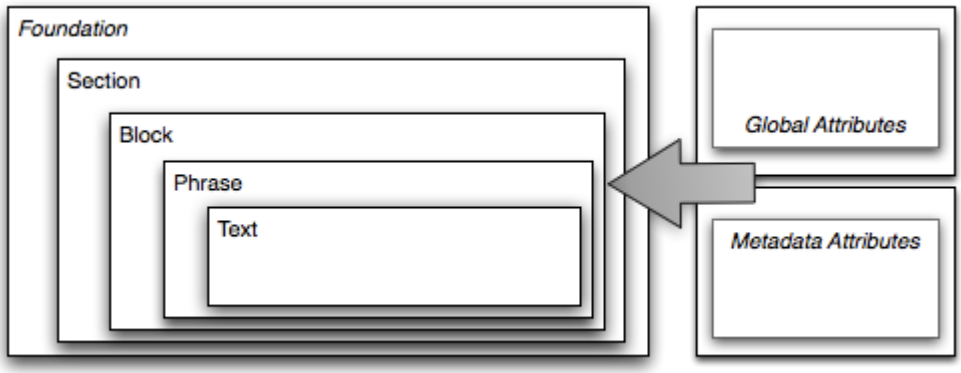

Figure 2: The document foundation and four layers of the Abstract Document Model

The layers of the Abstract Document Model are also how the model moves from abstraction of a document into formalization of the concept; they are more than just a way of conceiving of a document, but also represent the practical implementation of each profile through schema classes. Profile creators must not change the document foundation, nor remove layers or their inherent traits. The ability to create new profiles to describe information resources lies in the freedom offered within each layer to include elements and attributes to create new markup models. As a result, the Abstract Document Model can be used to create schematic definitions for documents across a wide spectrum of formats and fields while ensuring a predictable core structure.

The Z39.98-AI specification was built on this model specifically to address the problem of extensibility that existed in earlier versions of the Z39.86 standard. No single schema can encompass all document types and provide all the structure and flexibility necessary to generate all accessible formats. Although extension points were introduced in previous versions of Z39.86 to try and address this problem, they only allowed authors fixed entry points into a fixed model; the ability to fully recompose a schema to accommodate a new form was lacking.

The Abstract Document Model instead provides a common framework and over-arching set of rules in which profiles can be built without imposing unnecessary production and semantic restrictions. With the ability to insert new components at each layer, and at the same time leverage the components available in the core modules and from external features, the Z39.98-AI specification unshackles production from the monolithic approach of pre-defined and rigid markup models.

4.2 Fundamentals

4.2.1 Document foundation

At the root of the Abstract Document Model is the document foundation. The foundation, as its name implies, defines the concrete concept of a document and provides the practical containers through which the layers operate to define information resources.

The document foundation also represents the universal concept of a Z39.98-AI document: as composed of a container for metadata and other informational resources and a container to express the body content of the document. The foundation and this principle are constants across all profile implementations.

Example 1 shows how the document foundation is practically expressed in XML markup in all conformant profiles.

Example 1: Document foundation as an XML document

```
<document xmlns="http://www.daisy.org/ns/z3998/authoring/">
  <head>
    …
  </head>
  <body>
    …
  </body>
</document>
```

4.2.2 Document structure layers

The Abstract Document Model comprises a hierarchy of four layers that define a document's internal structure from its most general sense to its most concrete:

- the Section layer defines the primary structural divisions within documents;
- the Block layer defines the major structural elements that occur within sections of documents;
- the Phrase layer defines lexical and grammatical constructs; and
- the Text layer defines the mixture of character data and character elements at the core of all the bounding layers.

The four structural layers of the Abstract Document Model are a change from previous versions of Z39.86 in that they formally lay out a set of common features for documents and add granularity to the concept of block and inline elements. The new Section layer abstracts higher-level document structure concepts and the Text layer formally separates character data from the "inline" concept, providing a finer-grained data modeling system in which to design and build profiles.

Example 2 depicts how this hierarchical relationship turns into practical markup.

Example 2: Document structure layers depicted as XML

```
<body>
    <section>
        <block>
            <phrase>text</phrase> text <phrase>text</phrase>.
        </block>
        <block>
            …
        </block>
        <section>
            …
        </section>
    </section>
</body>
```

Moving down through the Abstract Document Model, each layer is structurally subservient to the layers that contain it, meaning layers must not contain elements from a superordinate layer. Using the preceding example to illustrate, it would be invalid for either `block` element to contain a child `section` element, as the nature of the Block layer prevents its elements from representing the hierarchical structure of a document.

4.2.3 Document attribute collection

The Abstract Document Model also comprises two attribute collections that augment and supplement the information in the document structure layers:

- The Metadata Attributes collection defines a set of attributes for describing information about documents and inflecting structures with semantic meaning.
- The Global Attributes collection defines a set of common attributes that are available to all elements in a document.

These collections are not bound or constrained by the structural layers, but make their attributes available across all elements. (Individual layers and elements may introduce restrictions and requirements on their use, however.)

The Metadata Attributes collection provides the ability to attach metadata directly to any element in a document (for example, to distinguish the role of generalized elements). These attributes may be used both on elements in the body and to supplement metadata elements in the header.

Example 3: Example of metadata attributes

```
<section about="origin_of_species">
    <h property="dc:title">On the Origin of Species by Means of Natural
Selection</h>
    <p property="dc:creator">Charles Darwin</p>
</section>
```

The Global Attributes collection similarly provides the ability to append commonly-used attributes across the entire markup model. These attributes typically provide instructions and hints to processing agents, but any attribute may be added if it meets the criteria of universal applicability. (Attributes in this layer include the specially-defined XML attributes and attributes used for internationalization purposes.)

Example 4: Example of global attributes

```
<section xml:id="s1">
    <p xml:id="p1">
         <w xml:id="w1">…</w>
    </p>
</section>
```

4.2.4 Constraints

The Abstract Document Model's concept of hierarchical layer containment forms the basis for the following set of constraints when creating new Z39.98-AI profiles:

- *The* `document` *element must not be redefined.* Allowing the nature of the element to be changed would make the concept of a document a moving target.
- *All elements added by modules and features must conform to the content definition of an existing layer.* The Abstract Document Model must not be ignored and elements not created that invalidate it. Not all elements must be added to a layer's class, but each must conform to the definition of content for a layer as expressed in constraint #3. Layers also must not be changed or added, as such changes would make the model unstable and unpredictable.
- *Every element's nature must match the definition of content for its containing layer.* This rule ensures the hierarchical principle of the Abstract Document Model is respected by preventing the creation of elements in incorrect contexts (e.g., a quote element could not be created in the Section layer as a quote does not represent a structural division of a document). Refer to each layer's definition for more information about its nature.
- *Every element's content model must be consistent with nature of the layer it belongs to.* This rule requires that elements from superordinate layers not be allowed as content of subordinate elements, which would represent an obvious breach of the hierarchical nesting principle. A Phrase layer element, for example, must only reference other Phrase

or Text elements. If a layer does not contain a definition for a needed element, a new variant should be created (so long as it is consistent with that layer, as per constraint #3).

- *Default members must not be removed.* Each layer defines at least one default member, which is typically a semantically neutral element that embodies the nature of the layer in the Abstract Document Model (i.e., from which all other elements inherit their base nature). In some cases, a specialization may replace the default member (for example, a newspaper profile could replace the `section` element with an `article` element as the default member of the Section layer). Refer to the specialization rules in each layer's implementation section and in the component definition for each default member for more information.

  Although the default member must not be removed, it is not a requirement that all layers be utilized when building content models. A Block layer element, for example, could allow only other Block elements as children, only Phrase elements, or only Text layer character data and elements. Similarly, the element could allow a mix of content from any and all of those three layers.

  NOTE: This constraint also applies to the attribute sets, even though attributes do not inherit from their default members.

4.2.5 Classes

Each layer in the Abstract Document Model defines a class that represents the practical implementation of its abstract concepts. In schema terminology, classes are the equivalent of name classes [RelaxNG] and named groups [XML Schema Part1] and allow the grouping of elements and attributes into complete and partial content models.

Elements and attributes added to a class during module activation automatically become available to all other elements that reference that class in their content model. Classes consequently can be viewed as aggregators and containers of a layer's traits (but do not define traits).

New subclasses may be created to define element and attribute models so long as the new subclasses are consistent with the content rules for their parent layer or collection. All new subclasses must be prefixed by their primary class name using dot notation (for example, a restricted set of Block layer elements for use including prefatory material could be named `Block.Intro`).

4.3 Layer and Collection Definitions

4.3.1 Introduction

The following sections detail the nature and function of each component of the Abstract Document Model, and introduce the rules and constraints inherent to them and that apply when adding new elements and attributes.

When creating a Z39.98-AI feature, the structures available for use in a host grammar must comply with the membership constraints of the layer or set to which the structures are contributed.

Note: All classes introduced in the following definitions are defined in the global-classes module definition.

4.3.2 Document foundation

The root element for all Z39.98-AI documents is the `document` element, which must contain the following two children:

- the `head` element – the child container for document information
- the `body` element – the child container for document content

This structure must not be altered by profile creators, although attributes may be added to these three elements. Profile creators may modify and/or extend each of the child containers. (Refer to the component definition for the `document` element for complete information.)

When modifying the information container, profile creators must ensure that it does not allow content that must be rendered by processing agents as document content. The information container should contain only metadata describing the document and informational matter that expands on, or describes, body content (e.g., definitions of terms, expansion of abbreviations, etc.).

All profiles must allow multiple instances of the `meta` element in the information container in order to identify the profile a document conforms to (see 8.3, *Referencing profiles and features*) and to allow expression of required metadata (see 8.5.2, *Required document-level metadata*).

As the document content container will typically be structured into parts, chapters, and sections, its default nature is to allow members of the Section layer. The `body` element may be treated as a specialization of the `section` element for cases where it will provide the only structure (i.e., contain only Block layer content).

4.3.3 Section layer

4.3.3.1 Description

The Section layer defines structurally-significant document divisions.

Structural significance is defined as the high-level grouping of content according to industry-accepted conventions for divisions, such as defined in the *Chicago Manual of Style* [CMoS], and by grouping according to sequentially-related or subordinate headings (real and implied).

Examples include: front, body, and back matter divisions; covers and other document bindings and containers; and major document divisions such as sections, parts, and chapters.

4.3.3.2 Implementation

Elements contributed to this layer are collectively referenced in a profile's `Section` class. The default member is the `section` element, whose component definition specifies the traits that all elements contributed to the layer must inherit. The `section` element may be replaced by a specialization so long as the specialization exhibits the same traits laid out in the component definition.

The `Section` class may be extended to include other elements, but these must adhere to the following rules:

1. They must not allow more than one occurrence of a structural heading in their content models. If an element contains headings that are not significant to the overall structure of the document, another distinct non-structural heading type must be used.
2. They must either contain exclusively Block or Section layer content or a mixture of the two. Phrase and Text layer content must not be allowed as direct descendants.

4.3.4 Block layer

4.3.4.1 Description

The Block layer contains structures and divisions that complement, and are subordinate to, the structurally significant divisions of the Section layer.

Block content differs from Section content in that it typically encapsulates information for a very specific component of a document or groups content for semantic or formatting reasons only. Block content likewise differs from Phrase content in that it establishes a connection between content ordered and divided vertically on a rendered page.

Examples include: headings, tables, lists, figures, quotes, and paragraphs.

4.3.4.2 Implementation

Elements contributed to this layer are collectively referenced in a profile's `Block` class. The default member is the `block` element, whose component definition specifies the traits that all elements contributed to the layer must inherit. The `block` element may be replaced by a specialization so long as the specialization exhibits the same traits laid out in the component definition.

The `Block` class may be extended to include other elements, but these must adhere to the following rules:

1. They must not allow structural headings within their content models. Structural headings are reserved for use in Section layer elements.

2. They must either contain exclusively Text, Phrase, or Block layer content or a mixture of the three.

4.3.5 Phrase layer

4.3.5.1 Description

The Phrase layer contains grammatical and other semantically significant segments that form the content of documents.

Phrase content differs from Block content in that it establishes a connection between content that flows horizontally across a rendered page. It also differs from Text content in that it does not define character data but operates at a grammatical and lexical level.

Examples include: sentences, terms, and words.

4.3.5.2 Implementation

Elements contributed to this layer are collectively referenced in a profile's `Phrase` class. The default member is the `span` element, whose component definition specifies the traits that all elements contributed to the layer must inherit. The `span` element may be replaced by a specialization so long as the specialization exhibits the same traits laid out in the component definition.

The `Phrase` class may be extended to include other elements, but these must either contain exclusively Text or Phrase layer content or a mixture of the two.

4.3.6 Text layer

4.3.6.1 Description

Although the Text layer's primary role is to define character data, it may also be extended to include elements to supplement and augment the available set of characters.

Text layer content differs from Phrase content in that it only represents character data and formatting, and does not infer any semantic information about itself.

Examples include: emphasis, superscripts, and subscripts.

4.3.6.2 Implementation

Elements contributed to this layer are collectively referenced in a profile's `Text` class. The Text layer is unique in that it does not define an element as having default membership but instead designates character data as its default member (as defined in the Characters section in [XML]).

All elements that incorporate character data in their content models should do so either by directly referencing the `Text` class or by indirectly referencing it through their layer's class

definition (if the layer incorporates character data). Custom character data declarations may be used where required.

The Abstract Document Model imposes no further restrictions on the allowed character content than is already defined by the Characters definition [XML]. Individual profiles and features may express additional restrictions on character content, but only within the scope of the elements and attributes that they define.

The `Text` class may be extended to allow expressions that are not supported by Unicode or that must be handled specially by processing agents (for example, for braille rendering). All extensions must allow only character data and other Text layer elements (refer to the `char` element for an example).

4.3.7 Metadata Attributes collection

4.3.7.1 Description

The Metadata Attributes collection provides the means to annotate documents and structures with meta information relating to their nature or semantics.

4.3.7.2 Implementation

Attributes contributed to this layer are collectively referenced in a profile's `Meta` class. This class has two default members:

- the RDFa attribute set; and
- the role attribute.

The RDFa attributes may be used throughout documents to append machine-readable RDF metadata to elements, while the `role` attribute may be used to layer specific semantic information about the nature of an element.

The `Meta` class may be extended to include other attributes, but these must adhere to the following rules:

1. They must only attach or inflect meta information.
2. They must not be used in contexts where their contents may be interpreted as document content.

For more information on the use of metadata in Z39.98-AI documents refer to 8.5, *Metadata*.

4.3.8 Global Attributes Collection

4.3.8.1 Description

The Global Attributes collection makes its member attributes available to all elements in a document. The members of this set do not share a common nature or purpose beyond their global applicability, however.

4.3.8.2 Implementation

Attributes contributed to this layer are collectively referenced in a profile's `Global Attributes` class. This class has two default members:

- the XML special attributes; and
- the Internationalization (i18n) attributes.

The `Global Attributes` class may be extended to include other attributes. There are no restrictions on the types of attributes that may be added to this class, but attributes that serve metadata roles should be added to the `Meta` class (see 4.3.7, *Metadata Attributes collection*).

All elements added to the Abstract Document Model classes should include the attributes defined in this layer in their attribute models.

5 Modules

5.1 Introduction

Modules are the high-level building blocks of Z39.98-AI profiles and together with features form the basis through which new markup models are created. Modules may be pre-defined (as in the case of the core modules), may be composed by profile authors to fit specific structural needs or may be made available for public use by interest groups.

Modules, in their practical form, are schema files that define custom components—such as elements and attributes—for use in building profiles. The components within any module typically share a common nature or serve a common structural purpose, allowing the targeted activation of modules for specific uses. This approach to module definition enables element and attribute counts to be kept to a minimum when composing new grammars.

5.2 Module and component definitions

Module and component definitions are informational resources that define the nature of a given module and its components. These definitions provide the default state of modules and components prior to activation, as well as high-level instruction on their proper use.

A definition must contain the module name, its dependencies, and a description of its purpose. Each component defined in the module must also be listed along with usage information, as outlined in 5.3, *Components*. Core modules additionally specify whether the module is required to be activated in all Z39.98-AI compliant profiles.

There are no rules on the format and structure of a definition resource. So long as complete information about the nature and traits of each module and component is made available in human-readable form, the requirement to provide a definition has been satisfied.

Definitions must be provided for all module components created specifically for use in the Z39.98-AI Framework, and each profile's resource directory must indicate the location of these definitions. Core module definitions are available at http://www.daisy.org/z3998/2012/auth/cm/.

Definitions for imported features are recommended but are not a requirement, as the official documentation maintained by the authoring body should be referenced instead.

Note that module and component definitions are not a reference to how to mark up a document to conform to a specific profile and should never be consulted to resolve validation issues. The normative schema available in the profile's resource directory is the authoritative reference to creating valid documents.

5.3 Components

5.3.1 Outline

A component is any single element, attribute, datatype, set of values, or pattern defined in a module file that may be activated and used to build content models. Components may be usable individually or may be dependent on parental or child relationships also being fulfilled. Some components may also have dependencies on components from other modules.

Each module definition must provide the following usage information for all its components:

- a semantic definition;
- a default usage context;
- a default content model;
- a default attribute model;
- a definition of alterability; and
- a definition of optionality.

5.3.2 Semantic definition

The semantic definition provides a concise statement about the kinds of structures or data the component is meant to represent.

The semantic statement may be very restrictive about what the component can capture (as in the case of highly-specialized components like tables of contents) or may be very open (as in the case of general containers).

The semantic definition must always be respected when including and using components in a Z39.98-AI profile; it is not valid to use a component in a way that contradicts its definition.

5.3.3 Default usage context

The default usage context identifies how the component is initially configured to be used within the Abstract Document Model and/or within other components:

- If the component references classes in the Abstract Document Model, then the default parent of the element or attribute is any element that allows the referenced class in its content or attribute model, respectively.
- If the component has a single element as its parent, then the referenced element is the default parent of the element or attribute.

Elements and attributes have an alterable usage context unless they are explicitly declared to be fixed. When altering a usage context, elements must not be moved across layer boundaries in the Abstract Document Model.

The default usage context must be declared using prose and/or expressions in RNC syntax in the module's definition.

5.3.4 Default content model

Each element and attribute in a module must declare a default content model or datatype that specifies its allowed content or value(s):

- In the case of elements, the default content model is a set of references to classes, elements, and/or datatype declarations.
- In the case of attributes, a datatype must be specified, which may reference other datatype declarations and/or enumerated literal values to set the initially-allowed value(s).

Every element or attribute has an alterable content model unless it is explicitly declared to be fixed. Declarations of fixed content models may apply to the content model as a whole, or to individual components of it.

The default content model must be declared using prose and/or expressions in RNC syntax in the module's definition.

5.3.5 Default attribute model

In addition to their default content model, elements must additionally declare an attribute model that specifies a set of references to the attribute classes and/or individual attribute declarations that constitute its default set of allowed attributes.

Every element has an alterable attribute model unless it is explicitly declared to be fixed. Declarations of fixed attribute models may apply to the attribute model as a whole, or to individual attributes within it.

All elements should allow the Global Attributes collection by default.

The default attribute model must be declared using prose and/or expressions in RNC syntax in the module's definition.

5.3.6 Definition of alterability

Each component's default content model, attribute model, and context are open to alteration when its containing module is activated unless the component's definition contains explicit restrictions stating otherwise. The semantic definition and rules of alterability and optionality are fixed and must not be changed.

Alterations to the default content and attribute models may be made to either restrict or loosen their allowed content/values. Complete rewrites of these initial states may also be permissible.

The default usage context may likewise be restricted, loosened, or rewritten to change the way that the component can be used by other components and within layers of the Abstract Document Model.

Although alteration of a component's traits is allowed by default, alterations must not be made that would cause the nature of the element to conflict with its semantic definition.

5.3.7 Definition of optionality

As modules may contain more components than are needed in the Z39.98-AI profile into which they are activated, any of their components may be specified as optional to prevent unnecessary imports.

A component must not be excluded if another included component requires it as a dependency.

By default, all components are optional; if a component is mandatory, this must be explicitly stated in the component's definition.

5.4 Expression

A module is formally expressed using one or more normative schemas authored in any of the languages specified in Appendix B, *Schema languages*.

5.5 Activation

Activation of a module is the process of importing it into a Z39.98-AI profile and including its components in content models and class definitions The first step in the activation process is to import the desired module into a profile driver file. If the module lists other modules as dependencies, those modules must also be imported, including any of their dependencies. This process of importing module dependencies continues until all required modules have been included. (Note that each dependency only needs to be imported once.)

Modules may have both hard dependencies, where failing to import a dependent module will result in a schema error, and soft dependencies, where no errors are reported. Omitting a soft dependency, while not resulting in an error in the schema file, still results in an invalid Z39.98-AI profile.

The next step in the activation process involves addressing the usage requirements for each component in the module, namely:

- whether the component is needed or can be removed if allowed by the component's optionality trait;
- whether the default content model and/or attribute model adequately reflect the needs of the profile being built and customizing the model if alterations are permitted; and
- whether the default usage context is appropriate for the profile being built and modifying where the component can be used if alterations are permitted. In the case where a component is needed in another layer of the Abstract Document Model, a new variant would need to be created.

Module files should not be directly edited to make modifications; any changes should be made in the profile driver file where the module is included. Example 5 shows how a definition can be overridden when activating a module.

Example 5: Modification of an RNG module during activation

```
<include href="./mod/z3998-document.rng">
  <!-- redefine body content to use partitions -->
  <define name="z3998.body.content">
     <ref name="z3998.partitions"/>
  </define>
</include>
```

Altering the components of a module is not required, and in some cases may not be permitted. If a module is fixed and not alterable, or if all of its components are to be used in their default state, the activation process may end with the import of the module.

The activation process must be carried out for all the required modules, optional modules, and features needed to fully represent the new Z39.98-AI profile.

5.6 Core modules

The core modules were developed to accompany this specification to assist in the creation of new profiles. Some of the modules are required to be activated in all compliant profiles, as they define the document foundation and other core aspects of the Abstract Document Model, while the other optional modules provide a common base of components to encourage structural consistency across profiles.

The modules are grouped so that each contains a single set of semantically- and/or structurally-related elements and attributes. The definition for each module can be found in the ANSI/NISO Z39.98-2012, *Core Modules* document that accompanies this specification. Profile creators should always check these definitions for compatible components before constructing a new module.

Core modules must not be directly modified to alter their component definitions. If module components allow modifications to any of their traits, those modifications must be made in the profile driver file during activation.

Profile creators are advised not to reference the publicly available versions of the core modules in their profile driver files, as the public location always contains the latest releases. The core modules are subject to change independent of this specification, so local copies of the modules should always be referenced to ensure the stability of any profile.

5.6.1 Core namespace URI

The Z39.98-AI Core namespace URI is: http://www.daisy.org/ns/z3998/authoring/

This is the default namespace URI of all elements defined in the core modules.

5.6.2 Core modules URI

The canonical URI of the Z39.98-AI core modules is: http://www.daisy.org/z3998/2012/auth/cm/

The current definitions for all core modules are available at this URI.

6 Profiles

6.1 Introduction

A profile is the collection of resources that together define the rules and requirements for marking up an information resource. Profiles are typically equated with XML schemas, but a schema may be only one of a number of resources that define the requirements; the prose definition that accompanies each module may outline implicit attribute values that must be recognized when an attribute is not present, for example, and an RDF vocabulary may define a set of allowed role values for adding semantic information to elements. It is also not uncommon that more than one schema may define the markup requirements: a RelaxNG schema to express the general structure of a document and a Schematron schema to express finer-grained conformance logic.

Where the Abstract Document Model defines Z39.98-AI document structure at a conceptual level and, coupled with the core modules, provides a common framework of composition rules and abstract building blocks, profiles define the practical markup models suitable for describing information resources. Examples of the types of resources that can be defined using the Framework include:

- general print books of all kinds (fiction and nonfiction)
- student textbooks and learning materials
- poetry
- plays
- newspapers
- periodicals
- bills and invoices
- catalogs
- consumer medical information

This specification does not place any restrictions on the scope of profiles; depending on their intended use, profiles may have a wide scope (such as a general book profile) or they may have a narrow scope (such as a consumer medical information profile).

6.2 Creation

The process of creating a profile typically begins with the authoring of a schema to represent the desired information resource(s). The Z39.98-AI Framework encourages a plug-in model for creating schemas, whereby existing modules and features are reused so that only the specialized requirements of the profile result in new elements and attributes.

A thorough understanding of the requirements of this specification will be necessary to successfully create a new schema. Compliance with the Abstract Document Model, for example,

requires that all profiles must first include the document module in their schema driver file (the schema file that will be referenced by validators) to establish the root `document` element. The global-classes module must then be imported to establish the Abstract Document Model classes into which all other module and feature components will be added.

This process of dependency and required component inclusion continues until a complete markup model is established that conforms to the requirements outlined in 6.3.2, *Profile markup model definition* (see also 5.5, *Activation* for more information on the inclusion process). It is also during this process that other external resources, such as RDF vocabularies, will be identified and incorporated into the profile.

The next step in creating a profile is to determine whether, in fact, a new unique profile has been created. The uniqueness of a profile is not determined solely by the modules and features that it includes, however; it is possible for many unique profiles to be created using the same core modules. The actual distinction between profiles is determined by variations across all of the following key factors:

- which modules have been activated;
- the number of alterations made to the components of the activated modules;
- the number of new elements (specializations) introduced;
- which features are defined as supported by the profile;
- the RDF vocabularies supported; and
- the restrictions, requirements, implicit values, and other conditions spelled out in any accompanying normative prose.

When the new profile is ready for use, a resource directory must be published at the schema's identity URI (see 6.3.1, *Profile identity URI*). This document identifies the schema files, RDF vocabularies, documentation, and other associated resources in a manner that allows humans and machines to locate them.

If a profile is being created to serve only a unique subset of a larger existing profile (or to provide a localization), the documentation that accompanies it should clearly identify its parent model as well as where it varies to assist processing agent developers.

The Z39.98-AI Working Group developed a set of profiles that conform to this specification. References to these profiles can be found in A.1, *Profile catalog*. While the intent is that this profile catalog will be extended over time, the profiles listed in the catalog are not an exclusive set; other agencies may create profiles and, if desired, make those available independently for public use.

6.3 Profile conformance definition

6.3.1 Profile identity URI

Each profile must define an identity URI that serves as its unique identifier.

An example profile identity URI is:
`http://www.example.org/z3998/2012/auth/profiles/myProfile/1.0/`

The identity URI must resolve to a resource directory where the profile is defined.

The provider of the feature or profile owns the identity URI and must ensure that the identity URI has no public access restrictions and is valid with respect to [RFC3986].

When comparing two identity URIs for equality, processing agents must follow the rules specified by [RFC3986].

For information on how a Profile Identity URI is expressed in Z39.98-AI documents, see 8.3, *Referencing profiles and features*.

6.3.2 Profile markup model definition

Each profile must define exactly one markup model, which must meet all of the following criteria:

- The grammatical constraints of the markup model must be expressed formally using at least one schema, identified in the profile's resource directory as normative and written in one of the languages identified in Appendix B, *Schema languages*.
- Grammatical constraints that cannot be expressed using schema syntax must be defined using prose in the designated area of the profile's resource directory.
- The schema identified as normative must adhere to the Abstract Document Model constraints defined in 4.2.4, *Constraints* and in each Implementation section in 4.3, *Layer and Collection Definitions*.
- If the profile supports one or several features, then the profile's normative schema must include the activated normative schema fragment(s) defined by these features.
- The activation of a feature's schema fragment must be done without violating the definition of available components in the feature's resource directory.
- Profiles must not modify the grammatical constraints expressed by the feature's schema fragment(s) except where allowed by the feature's normative specification.
- The grammatical constraints of the markup model must allow the expression of required metadata as defined in 8.5.2, *Required document-level metadata*.
- The grammatical constraints of the markup model must allow the identification of the profile to which the markup model conforms as well as any features in use, as defined in 8.3, *Referencing profiles and features*.

- The grammatical constraints of the markup model must allow the expression of RDF vocabulary associations defined in 11.3, *Associating vocabularies with Z39.98-AI documents*.
- All elements included in the profiles markup model must be bound to an XML namespace.

6.3.3 Profile resource directory

The identity URI must resolve to a resource directory that meets all of the general criteria defined in *10.2, Resource directory conformance definition*. In addition, the profile's resource directory must meet all of the following criteria:

- The name and version used to identify the profile in Z39.98-AI documents must be specified using the z3998-instance-name and version terms, respectively.
- The normative schema(s) must be identified in the normative-schemata section.
- Any additional grammatical restrictions expressed in prose must be identified using the normative-prose term.
- The URI of the RDFa initial context document must be identified using the z3998-rdfa-context term.
- Features supported by the profile must be identified using the z3998-feature property. (Note that feature version information is available implicitly through the normative schemas of the given version of the profile.)

6.3.4 RDFa initial context

The initial context document for each Z39.98-AI profile must meet all of the following criteria:

- It must be marked up in XHTML+RDFa 1.1 and be valid to the requirements for initial context documents stipulated in section 9 of [RDFa].
- It must not define more than one default vocabulary using the `vocabulary` term defined in [RDFa].
- The profile must reserve the following prefixes for the following URIs:

  z3998
  : `http://www.daisy.org/z3998/2012/vocab/decl/#`

    The Z39.98-2012 Instance Metadata Vocabulary.

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

    The Dublin Core Metadata Element Set, Version 1.1.

7 Features

7.1 Introduction

Certain kinds of content structures have highly specialized requirements for representation and processing, but have limited scope within information resources. A typical example is mathematical equations, where there already exists a specialized markup to represent them [MathML]. Z39.98-AI features are designed to handle these kinds of structures.

Features have much in common with profiles: each feature defines a markup model and has a unique URI associated with it that serves as its identifier and that resolves to a unique resource directory. Features are more specialized, however, and narrower in scope; they are not intended to be used as complete markup languages on their own. In the Z39.98-AI framework, features may be optional add-ons to profiles.

Features typically only expose a small set of elements and attributes for reference within the content models defined by profiles. These components are defined by the feature in its associated normative prose in order to ensure consistency of usage across Z39.98-AI profiles. The predictability of features further allows processing agents to support them independently of the profiles that might reference them.

Profiles explicitly declare which (if any) features they support. If a profile supports a feature, it becomes a "host" of this feature, and the profile's normative schema includes the schema fragment defined by the feature.

7.2 Feature conformance definition

7.2.1 Feature identity URI

Each feature must define an identity URI that serves as its unique identifier.

An example feature identity URI is:
`http://www.example.org/z3998/2012/auth/features/myFeature/1.0/`

The identity URI must resolve to a resource directory where the feature is defined.

The provider of the feature owns the identity URI and must ensure that the identity URI has no public access restrictions and is valid with respect to [RFC3986].

When comparing two identity URIs for equality, processing agents must follow the rules specified by [RFC3986].

For information on how Feature Identity URIs are expressed in Z39.98-AI documents, see 8.3, *Referencing profiles and features*.

7.2.2 Feature markup model definition

Each feature's markup model must meet all of the following criteria:

- Its grammatical constraints must be expressed formally using at least one schema or schema fragment, each of which must be written in one of the languages identified in Appendix B, *Schema languages* and be identified in the feature's resource directory as normative.

  Grammatical constraints that cannot be expressed using schema syntax must be defined using prose in the designated area of the feature's resource directory.
- It must not include contributions that subset or reduce a host profile's markup model. Features must only extend the host markup model.
- All elements it contributes to a host profile's markup model must be bound to an XML namespace.

  The namespace of elements and attributes defined by a feature must not be the Z39.98-AI Core namespace URI. Note, however, that a feature may reuse elements in the Z39.98-AI Core namespace in internal content models.
- It should allow the Metadata Attributes collection on the elements it contributes. Any deviations from this recommendation must be explicitly expressed in the feature's resource directory.
- It should allow the Global Attributes collection on the elements it contributes, but the use of local equivalents is allowed where contextually necessary to avoid naming collisions and/or redefinition of existing attributes (for example, to avoid the duplication of ID attributes). Any deviations from this recommendation must be explicitly expressed in the feature's resource directory.
- It must identify which components are available for use in a host grammar. Only the designated elements, attributes, values, and datatypes may be referenced directly by a profile in its markup model. (See *7.2.3, Feature resource directory*.)

7.2.3 Feature resource directory

The identity URI must resolve to a resource directory that meets all of the general criteria defined in *10.2, Resource directory conformance definition*. In addition, the feature's resource directory must meet all of the following criteria:

- The name and version used to identify the feature in Z39.98-AI documents must be specified using the z3998-instance-name and version terms, respectively.
- The normative schema or schema fragments(s) must be identified in the normative-schemata section.

  Any additional grammatical restrictions expressed in prose must be identified using the normative-prose term.

- The identification of components available for inclusion in a host profile must be made in normative prose.
- If the feature defines processing agent behaviors that modifies the default behavior with regards to initialization, then those behaviors must be detailed in normative prose.
- The components contributed by the feature should include component definitions.

8 Documents

8.1 Introduction

The purpose of authoring Z39.98-AI profiles is ultimately to allow the creation of documents that represent information resources. A Z39.98-AI document is a well-formed XML document that conforms to all the normative constraints defined by a profile and any included features.

Although Z39.98-AI documents may exist as single XML files that reference no additional local or remote resources, a complete Z39.98-AI document set typically will include local resources (such as included XML fragments, images, style sheets, and metadata records), as well as reference remote resources that can be obtained by their protocol (web-hosted resources available over HTTP or FTP, for example). To simplify the distribution of documents, the Z39.98-AI Framework consequently also defines a container format specification for the bundling of local resources with their associated Z39.98-AI document.

8.2 Document conformance definition

In order to be conformant, a Z39.98-AI document must meet all of the following criteria:

- It must be a well-formed XML Document as defined by [XML].
- It must be a well-formed namespace as defined by [XMLNAMES].
- It must constitute a single XML Infoset as defined in [XMLInfoset].
- The start tag of the root element of the document must explicitly contain an [XMLNAMES] compliant declaration for the Z39.98-AI Core namespace, as defined in Z39.98-AI Core namespace URI.
- It must include a reference to exactly one conforming profile, using the mechanism defined in 8.3, *Referencing profiles and features*.
- If it includes markup contributed by any of the features supported by the referenced profile, it must include a reference to those features used and only to those used using the mechanism defined in 8.3, *Referencing profiles and features*.
- It must be valid to the normative schema(s) of the referenced profile.
- It must conform to any normative prose expressed in the resource directories of the referenced profile and feature(s) (see 10.2, *Resource directory conformance definition*).

- It must contain metadata that meets all the requirements of 8.5.2, *Required document-level metadata*.
- It must perform RDF vocabulary association as defined in 11.3, *Associating vocabularies with Z39.98-AI documents*.
- If it references CSS Style sheets, then it should do so using the syntax defined in [XMLSTYLE].

8.3 Referencing profiles and features

Documents must reference the profile to which they adhere via a meta element in the metadata container element (see 4.3.2, *Document foundation*). The profile's identity URI must be given in the `resource` attribute and the value of the `rel` attribute must contain the term `profile` (as defined in the Z39.98-2012 Instance Metadata Vocabulary). The unique name and version number of the profile must also be identified using the `name` and `version` properties.

Each feature used in the document must also be referenced via a `meta` element in the metadata container. In referencing features, the `rel` attribute must contain the term `feature` (as defined in the Z39.98-2012 Instance Metadata Vocabulary). Each feature similarly must reference its unique name and version number.

Example 6 illustrates a compliant profile and feature reference in the `head` element of a Z39.98-AI document.

Example 6: Referencing a profile and features from a Z39.98-AI document

```
…
<head>
  <meta rel="z3998:profile"
resource="http://example.org/z3998/2012/auth/profiles/sciencejournal/1.0/">
    <meta property="z3998:name" content="sciencejournal" />
    <meta property="z3998:version" content="1.0" />

    <meta rel="z3998:feature"
resource="http://example.org/z3998/2012/auth/features/physics/3.0/">
      <meta property="name" content="physics" />
      <meta property="version" content="3.0" />
    </meta>

    <meta rel="z3998:feature"
resource="http://example.org/z3998/2012/auth/features/chemistry/1.1/">
      <meta property="name" content="chemistry" />
      <meta property="version" content="1.1" />
    </meta>
  </meta>
</head>
…
```

Note that in Example 6, the feature reference `meta` elements are nested within the profile reference `meta` element. This is not a conformance criterion. The order in which the references appear, and whether or not they are nested within each other, is not significant.

8.4 Referencing RDF vocabularies

Documents must include a `meta` element in the document `head` that references the URI of their initial context document, as illustrated in Example 7.

Example 7: Referencing an RDFa initial context document

```
<document xmlns="http://www.daisy.org/ns/z3998/authoring/" xml:lang="en">
  <head>
    <meta rel="z3998:rdfa-context"
resource="http://www.daisy.org/z3998/2012/vocab/context/default/" />
  </head>
  …
</document>
```

The referenced context document may define a default vocabulary, which allows the terms from that vocabulary to be added in their unprefixed form. A document that has the Z39.98-2012 Structural Semantics Vocabulary as its default, for example, would allow the use of the `chapter` term from that vocabulary as follows:

Example 8: Referencing a term from the default vocabulary

```
<section role="chapter">
  …
</section>
```

The referenced RDFa context document may also define one or more fixed prefixes for referencing terms from additional vocabularies. Example 9 shows the new markup that would result if the Z39.98-2012 Structural Semantics Vocabulary were instead associated with the prefix `struct`.

Example 9: Referencing a term from a prefixed vocabulary

```
<section role="struct:chapter">
  …
</section>
```

If additional vocabularies not defined in the RDFa context document are needed, the prefix attribute must be attached to the `document` element to declare them:

Example 10: Referencing an RDFa vocabulary

```
<document xmlns="http://www.daisy.org/ns/z3998/authoring/" xml:lang="en"
prefix="foaf: http://xmlns.com/foaf/0.1/">
  …
</document>
```

Note that the `xmlns` namespace declaration attribute [XMLNAMES] must not be used to declare vocabulary prefixes.

Example 11: Invalid vocabulary prefix declaration using the xmlns attribute

```
<document xmlns="http://www.daisy.org/ns/z3998/authoring/" xml:lang="en">
  <head>
    <meta property="dc:identifier" content="uid123" />
    …
  </head>
  …
</document>
```

8.5 Metadata

8.5.1 Introduction

Access to comprehensive metadata is critical to producers, distributors, and consumers of books and other digital resources. Fulfilling this need is complicated, however, by the wide variety of standards and applications of metadata. In response, the Z39.98-AI Framework has been created with an open approach to specifying and attaching information resources:

- it allows producers to reference multiple information resources, conforming to different standards, as their internal and external needs require, dropping the rigidity of embedded metadata;
- it fully embraces the RDF metadata standard and the `role` attribute for attaching machine-readable and grammar-definable metadata, ensuring that metadata has first-class standing within the Framework and can be easily attached where needed and as needed; and
- it incorporates metadata elements and attributes that allow for self-documentation of Framework components, simplifying the process of creating and publishing new profiles.

The rationale for referencing document metadata resources instead of embedding is two-fold. First, it allows the variety needed to meet the real-world production needs of producers. Second, using the Z39.98-AI document itself as the source of metadata is not an intuitively sound design principle; information retrieval, processing, and sharing are all greatly enhanced when metadata information remains distinct from the object it describes.

The move to standardized metadata attributes brings the specification in line with current best practices for attaching metadata. It also brings a measure of consistency to metadata by allowing controlled vocabularies to be defined (i.e., semantics are not purely at the discretion of producers).

There is no imperative that the complete document metadata actually appear in the document itself, so long as the document can be unambiguously identified and include information on how to obtain additional metadata. With that information, a processing agent can access whatever other information might be necessary—if any—to complete its work.

The metadata requirements for a Z39.98-AI document serve two purposes:

1. to uniquely identify the document and its publisher; and
2. to provide references to one or more resources that provide more detailed metadata.

Z39.98-AI documents use mandated Dublin Core metadata to identify the document and its publisher. Additional metadata is referenced by URI using constructs defined by an RDF vocabulary. The method of linking to metadata encompasses strategies as simple as a self-reference to the document itself (if the producer chooses to include metadata within the document), or as complex as URIs to web services that allow the querying of producer databases directly. Multiple complementary metadata may be referenced, allowing for different metadata standards to be used by different kinds of systems.

8.5.2 Required document-level metadata

Every conforming Z39.98-AI document must include a minimal set of metadata, defined below, that uniquely identifies the document and its publisher. Metadata is referenced using `meta` elements within the `head` element of the document. Metadata items are given as name-value pairs. The `property` attribute is used to identify the name of the metadata item and the `content` attribute, or alternatively `meta` element content, contains the actual metadata item value.

The metadata names used are taken from the Dublin Core Metadata Initiative [DCMES] vocabulary. All `property` attribute values must reference properties in the [DCMES] vocabulary using the `dc` prefix (see 8.4, *Referencing RDF vocabularies* for more information on associating vocabularies).

Every conforming Z39.98-AI document must contain the following three metadata items:

`identifier`
: A string that uniquely identifies the document as created by the document producer. Note that, in the case of a republisher, this identifier should not refer to the document source.

`publisher`
: The name of the publisher of this document.

`date`
: The date this document was last modified. For the purposes of Z39.98-AI documents, this value also serves as a version identifier. This property value must be a valid W3C `dateTime` of the form `CCYY-MM-DDThh:mm:ssZ`.

8.5.3 Document-level metadata resources

Z39.98-AI documents may refer to any number of metadata resources. These may include files (either local or remote), web services, or any other resource that can be referenced by a URI (including the Z39.98-AI document itself).

Metadata resources are identified using `meta` elements within the document `head` with the following required attributes:

`rel`
: Must refer to the meta-record term in the Z39.98-2012 Instance Metadata Vocabulary.

`resource`
: Must be a valid, resolving URI that points to the metadata resource.

Each metadata resource identified in the document must also have its type identified using an additional `meta` element with the following mandatory attributes:

`property`
: Must refer to the meta-record-type term in the Z39.98-2012 Instance Metadata Vocabulary.

`about`
: Must be the URI of the external metadata resource.

`content`
: Must refer to one of the terms in the Z39.98-2012 Instance Metadata Vocabulary that refer to metadata standards.

In addition, the specific version of the metadata type may be identified using a `meta` element with the following required attributes:

`property`
: Must refer to the meta-record-version term in the Z39.98-2012 Instance Metadata Vocabulary.

`about`
: Must be the URI of the external metadata resource.

`content`
: Must be a string identifying the metadata standard version.

Example 12 shows how metadata would be expressed in the `head` element of a Z39.98-AI document.

Example 12: Z39.98-AI document metadata

```
…
<head>
    …
    <meta property="dc:identifier" content="daisy-z2012-exemplar-01" />
    <meta property="dc:publisher" content="DAISY Consortium" />
    <meta property="dc:date" content="2012-07-27T18:50:05Z" />

    <meta rel="z3998:meta-record" resource="daisy-z2012-exemplar-01-
mods.xml">
        <meta property="z3998:meta-record-type" content="z3998:mods" />
        <meta property="z3998:meta-record-version" content="3.3" />
    </meta>

    <meta rel="z3998:meta-record" resource="daisy-z2012-exemplar-01-
onix.xml">
        <meta property="z3998:meta-record-type" content="z3998:onix-books" />
    </meta>

    <meta rel="z3998:meta-record"
resource="http://www.example.com/meta/generator?rec=exemplar-
01&amp;format=marc21">
        <meta property="z3998:meta-record-type" content="z3998:marc21-xml" />
    </meta>
    …
</head>
…
```

Note that the use of nested `meta` elements for external metadata resource references is not a conformance criterion. The order in which the references appear, and whether or not they are nested within each other, is insignificant.

8.5.4 RDFa metadata associations

Z39.98-AI documents may contain elements that function as micro-documents. The `article` element, for example, incorporates its own metadata container and content body, allowing it to function as a self-contained document. Typical [RDFa] processing of metadata expressions, however, requires that an expression that doesn't identify its subject (e.g., via an `about` attribute) "bubble up" to the nearest containing expression (see the concept of *chaining* in [RDFa]). If none is found, the expression represents some aspect of the document as a whole, such as its unique identifier or publisher.

In the case of micro-document elements, metadata without an explicit association ideally would similarly define some aspect of the containing micro-document. An RDFa processor will not make this association implicitly in the way that it does with the document, however, because the processor has no concept of containing elements. Consequently, it will either associate the expression with a property further up the document chain or with the document itself. Although the correct processing behavior, it is not what many authors will expect when adding metadata to a micro-document element.

To simplify the process of adding metadata to micro-documents, this specification introduces a special implied association mechanism for elements intended to function as micro-documents: the container element must be handled as though it has an `about` attribute attached that defines itself as the subject of all such expressions.

To exemplify, the following micro-document element declaration:

```
<article>
```

must be treated as equivalent to the following RDFa expression:

```
<article xml:id="article01" about="#article01">
```

By adding this implied association, all metadata defined within the container is ensured to have the micro-document as its subject if no other is defined.

Only elements that include both a metadata container and content body should be treated as micro-documents and enabling this behavior. If implicit associations are not desired, an element's documentation must explicitly state the lack of this supporting behavior.

In order to generate an accurate RDF graph from a document that relies on implied associations, a processing step to make all implicit associations explicit will need to be undertaken first, which may or may not also include adding missing `xml:id` attributes. Authors are encouraged to avoid relying on this implied mechanism, and the requisite intermediary processing, when RDF metadata generation and manipulation is an important workflow requirement.

8.5.5 Inline metadata

Metadata may also be expressed inline using the RDFa attributes available in the Metadata Attributes collection (i.e., on elements in the document `body`). This mechanism for specifying metadata may be used to provide either document-level metadata as outlined in 8.5, *Metadata* or to provide metadata for document fragments.

This specification does not mandate the use of any specific RDF vocabulary for inlining metadata information, but relies on the vocabulary association mechanism defined in 11.3, *Associating vocabularies with Z39.98-AI documents*.

Example 13 shows how to use inline metadata in the body of a Z39.98-AI document. The example assumes the Z39.98-2012 Structural Semantics Vocabulary has been declared as the default vocabulary for the document.

Example 13: Inline metadata

```
<section>
  <h property="fulltitle">
    <hpart role="title">On the Origin of Species by Means of Natural
Selection</hpart> or
    <hpart role="subtitle">the Preservation of Favoured races in the struggle
for life</hpart>.
  </h>
   …
</section>
```

9 Container

9.1 Introduction

The need to package Z39.98-AI documents together with the local resources they reference has always existed, but the method and format for achieving this goal has not been formally defined in previous versions of the specification. As a result, producers have had the freedom to package their document sets in any of a variety of formats as their needs required, while user agent developers have been left with the problem of handling this variation when processing the container files.

This specification seeks to redress this issue by formally defining the packaging format and MIME type [RFC2046] for bundling the local resources in Z39.98-AI document set, as well as to propose a common file extension format for the identification of container files. This approach will allow the rapid deployment and exchange of container files by removing the unnecessary ambiguity and investigation that has existed regarding their format and structure.

NOTE: Although remote resources are an integral part of a document set, they are referenced and obtained from their remote location by processing agents. Remote resources optionally can be obtained and included as local resources within the container for distribution/processing purposes, but any references to them in the Z39.98-AI document should be updated to point to their new local location to prevent repeat requests by processing agents.

9.2 Format

The Z39.98-AI container format uses a modified version of the Open Container Format [OCF] to bundle into a single file all the local resources that constitute a Z39.98-AI document set. All requirements detailed in the OCF specification apply to the creation of Z39.98-AI container files, except where modified by this section.

The Z39.98-AI container format employs the modified version of the ZIP format detailed in Section 4 of [OCF]. For identifying that the file is a Z39.98-AI container, however, the following MIME type [RFC2046] must be used instead: `application/z3998-auth+zip` (see *9.4, Media types*).

Each conformant container file must include a `META-INF` directory at the root of the container as per Section 3.5 of [OCF]. This directory must include the required `container.xml` file detailed in Section 3.5.1 of [OCF], which in turn must contain a `rootfile` element that points to the root Z39.98-AI document.

The root document must be identified by the MIME type `application/z3998-auth+xml` (see 9.4, *Media types*) and the container must include the root document as a resource. The container must not include references to external documents. If a document has been decomposed into smaller XML fragments, the root must be the XML file into which all the components are included to reconstitute the complete Z39.98-AI document.

The container must include the Z39.98-AI document together with all of the local resources associated with it. The container may include alternate renditions of a document, but must not contain more than a single Z39.98-AI document.

The optional reserved files allowed by the OCF specification are handled as follows:

- `metadata.xml`, `manifest.xml` and `signatures.xml` are not a normative part of the Z39.98-AI container format, but may be included if desired. Processing agent support must not be expected for these files or for the signing of container files, however.
- The contents of a Z39.98-AI container file must not be encrypted or have digital rights management schemes applied to them, so the `encryption.xml` and `rights.xml` files are not supported and should not be included. Where security is an issue, alternate means of protecting the container file should be pursued instead.

The Z39.98-AI container format does not support OEBPS package files [OPF] and their inclusion in container files is not recommended.

Example 14 shows the structure of a typical Z39.98-AI container file.

Example 14: Z39.98-AI container file structure

```
mimetype
META-INF/
   container.xml
Z3998-AI/
   On the Origin of Species.xml
   metadata/
      mods.xml
      onix.xml
   images/
      cover.png
```

Example 15 shows the contents of the `container.xml` file. The `rootfile` points to the root Z39.98-AI document.

Example 15: container.xml file mark up

```
<?xml version="1.0"?>
<container version="1.0"
xmlns="urn:oasis:names:tc:opendocument:xmlns:container">
  <rootfiles>
    <rootfile full-path="Z3998-AI/On the Origin of Species.xml" media-
type="application/z3998-auth+xml" />
  </rootfiles>
</container>
```

9.3 File extension

To facilitate the identification of the Z39.98-AI container files, containers should use the extension `.zai`. Alternate file extensions may be used so long as the container conforms with the ZIP requirements outlined in 9.2, *Format*.

File extensions are only a hint to the format of a container; the container's `mimetype` file is the primary designator of the contents.

9.4 Media types

This specification defines the following two new media types:

`application/z3998-auth+zip`
: identifies a Z39.98-AI container file.

`application/z3998-auth+xml`
: identifies a Z39.98-AI document.

Refer to Appendix C, *Media type registration* for more information.

10 Resource directories

10.1 Introduction

Profiles and features are fully-developed markup languages in their own rights, and typically require multiple documents to define them normatively: schemas, RDF vocabularies, and often normative prose. Profiles and features typically also provide informative resources, such as primers, best practice guides, and references to supporting software to assist users with their implementation.

To facilitate the discovery of these resources for both human and machine users alike, this specification uses [XHTML+RDFa] as the mechanism to associate a collection of normative and informative resources with a profile or a feature. The XHTML+RDFa document used to collect the references to these resources is called a resource directory.

As XHTML documents, resource directories are human-readable, and thus may serve as the primary source of information for all users of a profile or feature. Resource directories are also machine-readable documents, and thus may be sources for the discovery and retrieval of schemas, style sheets, RDF vocabularies, and other types of resources needed by processing agents.

In order to facilitate interoperability of processing agents, this specification defines a Z39.98-2012 Resource Directory Vocabulary. This vocabulary contains terms that describe and identify resources associated with profiles and features, and states how they are to be used.

10.2 Resource directory conformance definition

The resource directory for a profile or feature must be a document conforming to the [XHTML+RDFa] specification, using terms from the Z39.98-2012 Resource Directory Vocabulary as detailed below.

- The resource directory must include one or more references to relevant normative external specifications to which the profile or feature being described adheres. The term used to identify that resource must be specification-compliance.
- The resource directory must identify the designated maintenance agency of the profile or feature. The term used to identify that agency must be maintenance-agency.
- The resource directory should include information on the version history of the profile or feature. A reference to the most recent published version must be expressed using the current-version-uri term. A reference to a version history catalog must be expressed using the version-history term.
- The resource directory should include at least one resource identified by the term reference which resolves to a human-readable description of the profile's or feature's purpose and structure.

Note that additional resource directory requirements specific to profiles and features are defined in 6.3.3, *Profile resource directory* and 7.2.3, *Feature resource directory* respectively. All other terms defined by the Z39.98-2012 Resource Directory Vocabulary are optional.

This specification allows terms from other vocabularies than the Z39.98-2012 Resource Directory Vocabulary to be included in resource directory documents.

11 RDF vocabularies

11.1 Introduction

The Z39.98-AI Framework makes use of [RDFa] and the role attribute to express document-level metadata and to express semantic inflections on individual elements in the document content (see 4.3.7, *Metadata Attributes collection*). RDF vocabularies are used to define controlled lists of

terms to provide consistency in the application of metadata across profiles and between producers.

The Z39.98-AI Framework includes a set of predefined vocabularies (see *A.3, Vocabulary catalog*). This section describes the nature of these vocabularies and defines rules for how to associate these and additional vocabularies with a Z39.98-AI document. Rules for the processing of vocabulary terms are defined in 12.4, *Processing of vocabulary terms*.

11.2 Vocabularies defined by this specification

11.2.1 Z39.98-2012 Instance Metadata Vocabulary

This vocabulary defines a set of terms for use in declaring the profile to which the document conforms as well as any features in use (see 8.3, *Referencing profiles and features*). It also facilitates the expression of document-level metadata (see 8.5, *Metadata*). Use of this vocabulary is required in all profiles.

The canonical URI of the Instance Metadata Vocabulary is http://www.daisy.org/z3998/2012/vocab/decl/.

Document Instance Nature Properties

The following properties are used to declare the Z39.98-2012 document instance nature in terms of which profile it adheres to, which optional features it makes use of and which RDFa document defines its default context. The properties must be used in the rel attribute of the meta element in the document instance head.

profile

The associated resource reference is the canonical URI of the profile this document instance adheres to, also resolving to its Resource Directory.

feature

The associated resource reference is the canonical URI of a feature used in this document instance, also resolving to its Resource Directory.

rdfa-context

The associated resource reference is the default RDFa context for this document instance.

Profile and Feature Reference Properties

The following properties are used to fully identify a referenced profile or feature.

name

The name that uniquely identifies the referenced profile or feature.

version

The version number of the referenced profile or feature.

Metadata Record Reference Properties

The following properties are used when referencing external metadata records from a Z39.98-2012 document instance. The properties must be used on attributes of the meta element in the document instance head element.

meta-record

The referenced resource is a metadata record describing the document instance.

meta-record-type

The referenced metadata record is of the given type.

meta-record-version

The referenced metadata record is expressed using the given version of the given type.

Metadata Record Types

The following properties represent well-known metadata record types.

mods

The referenced metadata record is of the type Metadata Object Description Schema (MODS).

onix-books

The referenced metadata record is of the type ONIX for Books.

marc21-xml

The referenced metadata record is of the type MARC 21 XML.

dcterms-rdf

The referenced metadata record is an RDF XML record using properties from http://purl.org/dc/terms/.

dcterms-rdfa

The referenced metadata record is an RDFa document using properties from http://purl.org/dc/terms/.

11.2.2 Z39.98-2012 Structural Semantics Vocabulary

This vocabulary defines a set of terms relating to the description of structural semantics of documents in general, and written works in particular. It is primarily intended to be used for semantic inflection on individual elements in the document content. Use of this vocabulary is optional.

The canonical URI of the Structural Semantics Vocabulary http://www.daisy.org/z3998/2012/vocab/structure/.

Major document structures

Standard terms used for the classification, partitioning and dividing of a document.

Document types

fiction

A work of fiction.

non-fiction

A work of non-fiction.

article

A major independent section in a piece of writing, typically appearing in journals, magazines or newspapers.

essay

An essay.

textbook

A textbook.

catalogue

A catalogue.

Document partitions

frontmatter

Preliminary material to the content, such as tables of contents, dedications, etc.

bodymatter

The body content of a document.

backmatter

Ancillary material occurring after the document body, such as indices, appendices, etc..

Document divisions

volume

A component of a collection.

part

A major structural division of a piece of writing, typically encapsulating a set of related chapters.

chapter

A major structural division of a piece of writing.

subchapter

A major sub-division of a chapter.

division

A major structural division that may also appear as a substructure of a part (esp. in legislation).

section

A structural division typically subordinate in importance to a part or division (esp. in textbooks and legislation).

subsection

A major sub-division of a section.

Document sections

foreword

An introductory section that precedes the work, typically not written by the work's author.

preface

An introductory section that precedes the work, typically written by the work's author.

prologue

An introductory section that sets the background to a story, typically part of the narrative.

introduction

A section in the beginning of the work, typically introducing the reader to the scope or nature of the work's content

preamble

A section in the beginning of the work, typically containing introductory and/or explanatory prose regarding the scope or nature of the work's content

conclusion

An ending section that typically wraps up the work.

epilogue

A concluding section that is typically written from a later point in time than the main story, although still part of the narrative.

afterword

An ending section that typically details the history of the story or its significance.

Document reference sections

toc

A table of contents, typically appearing in the work's frontmatter, or at the beginning of a section.

appendix

Supplemental information.

glossary

An alphabetical list of terms in a particular domain of knowledge with the definitions for those terms.

bibliography

A list of works cited.

discography

A list of albums and/or songs by a performer.

filmography

A list of acting roles by a performer.

index

A detailed list, usually arranged alphabetically, of the specific information in a publication.

colophon

A brief description usually located at the end of a publication, describing production notes relevant to the edition.

Titles

title

The title of the work, or, when used in the context of a fulltitle, the primary part of the full title.

halftitle

The title appearing that appears on the first page of a work or immediately before the text.

Inherits from: title

fulltitle

The full title of the work, either simple, in which case it is identical to title, or compound, in which case it consists of a title and a subtitle.

Inherits from: title

Same as: http://purl.org/dc/terms/title

subtitle

An explanatory or alternate title for the work.

Inherits from: title

covertitle

The title of the work as displayed on the work's cover.

Inherits from: title

Preliminary sections and components

published-works

A list of additional works by the author.

title-page

The title page of the work.

halftitle-page

The half title page of the work.

acknowledgments

A passage containing acknowledgments to entities involved in the realization of the work.

imprint

Information relating to the publication or distribution of the work.

imprimatur

A formal statement authorizing the publication of the work.

loi

A listing of illustrations included in the work.

lot

A listing of tables included in the work.

publisher-address

The publisher's address, which can be abbreviated to city and postal code only.

publisher-address

A note from the publisher.

editorial-note

An explanation of an editor’s method, or a discussion of variant texts.

grant-acknowledgment

Describes financial assistance towards the work.

contributors

A list of contributors to the work.

other-credits

Acknowledgments of previously published parts of the work, illustration credits, and permission to quote from copyrighted material.

biographical-note

A biographical note about the author or authors, here including editors, compilers, and translators).

translator-note

An introductory note from the translator(s).

errata

Publication errata, in printed works typically a loose sheet inserted by hand; sometimes a bound page.

promotional-copy

A promotional description of the book, often containing quotations from reviews.

dedication

An inscription addressed to one or several particular person(s).

Document components

Non-structural components that occur within documents, such as letters, poems, quotations, etc.

pgroup

A group of interrelated paragraphs.

example

A sample, model, illustration or template representative of some aspect of the narrative.

epigraph

A quotation at the start of a document or section that typically introduces or sets the theme for what follows.

Annotations

annotation

Explanatory information about passages in the work.

introductory-note

An annotation whose purpose is to provide an introductory note.

Inherits from: annotation

commentary

An annotation whose purpose is to provide commentary.

Inherits from: annotation

clarification

An annotation whose purpose is to provide a clarification.

Inherits from: annotation

correction

An annotation whose purpose is to provide a correction.

Inherits from: annotation

alteration

An annotation whose purpose is to indicate an alteration made to the source document, such as the re-arrangement or removal of content.

Inherits from: annotation

presentation

An annotation whose purpose is to describe some aspect of the document layout, such as the formatting of a page or conventions used in rendering the content.

Inherits from: annotation

production

A temporary annotation not intended as part of the finished document, such as a production note

Inherits from: annotation

attribution

An annotation whose purpose is to provide a source or attribution.

Inherits from: annotation

Production roles

author

An author of a work.

editor

An editor of a work.

general-editor

The general editor of a work.

commentator

A commentator on a work.

translator

A translator of a work.

republisher

A republisher of a work.

Description types

structure

A structural description for a piece of content (e.g., a table structure summary).

Addresses

Physical

geographic

A positional location, typically expressed by coordinates.

postal

A postal or mailing address.

Virtual

email

An email address.

ftp

A file transfer protocol address.

http

A hypertext transfer protocol address.

ip

An internet protocol address.

Asides

aside

Secondary or supplementary content.

sidebar

Secondary or supplementary content, typically formatted as an inset or box.

Inherits from: aside

practice

A review exercise or sample.

Inherits from: aside

notice

A notification.

Inherits from: aside

warning

A warning.

Inherits from: aside

marginalia

Marginalia.

Inherits from: aside

help

Instructions/advice to the reader on using the work.

Inherits from: aside

Dialogue/Drama

drama

A work of fiction intended for performance.

scene

A structurally significant segment of a drama.

stage-direction

An instruction to an actor or director, included in the work.

dramatis-personae

A listing of the roles or characters appearing in the work.

Inherits from: collection

See also: persona

persona

The name or identity of an individual role or character appearing in the work.

See also: dramatis-personae

actor

The name of the actor who plays a character in the work.

See also: dramatis-personae

role-description

A description of a character in the work.

See also: dramatis-personae

speech

An utterance by a persona.

See also: persona

Diaries

diary

A diary.

diary-entry

A diary entry.

Figures

figure

An instance of a figure, which may include a title, caption and credits in addition to an image or table.

plate

A figure or image typically printed on a coated unnumbered page.

gallery

A section of a document containing more than one image, figure or plate. Gallery pages may be numbered or unnumbered, and on regular or special stock.

Letters

letter

An instance of a formal, informal or business letter.

sender

The name and address of the sender of a letter.

recipient

The name and address of the recipient of a letter.

salutation

An expression of greeting at the start of a letter.

valediction

An expression of farewell at the end of a letter.

postscript

An additional paragraph or note occurring after a signature in a letter.

Emails

email-message

An instance of a email message.

to

The recipient(s) email address line.

from

The sender's email address line.

cc

The carbon copy email address line.

bcc

The blind carbon copy email address line.

subject

The email subject line.

Lists

collection

A set of interrelated items.

orderedlist

A list where the order of the list's items is significant.

Inherits from: collection

unorderedlist

A list where the order of the list's items is not significant.

Inherits from: collection

abbreviations

A list of abbreviations.

Inherits from: unorderedlist

timeline

A chronological list of events

Inherits from: orderedlist

Notes

note

A footnote or a rearnote, linked from the main text via a note reference

See also: noteref

footnotes

A collection of notes appearing at the bottom of a page.

Inherits from: collection

See also: footnote

footnote

A note appearing at the bottom of a page.

Inherits from: note

See also: footnotes

rearnote

A note appearing in the rear (backmatter) of the work, or at the end of a section.

Inherits from: note

See also: rearnotes

rearnotes

A collection of notes appearing at the rear (backmatter) of the work, or at the end of a section.

Inherits from: collection

See also: rearnote

Verse

verse

A work typically composed of metered or rhyming lines (as opposed to prose).

poem

A poem.

Inherits from: verse

song

A song.

Inherits from: verse

hymn

A hymn.

Inherits from: song

lyrics

The text of a song.

Inherits from: verse

See also: song

Document text

Terms for describing text elements.

text

A segment of text.

phrase

A group of lexical words, comprising a single unit within a sentence.

Inherits from: collection and text

keyword

A key word or phrase.

sentence

A group of lexical words comprising a lexical sentence.

Inherits from: phrase

topic-sentence

A phrase or sentence serving as an introductory summary of the containing paragraph.

concluding-sentence

A phrase or sentence serving as a concluding summary of the containing paragraph.

t-form

An informal second-person pronoun.

Inherits from: phrase

v-form

A formal second-person pronoun.

Inherits from: phrase

Abbreviations

acronym

An abbreviation formed from initial letters of a name or expression, pronounced as a word.

initialism

An abbreviation formed from initial letters of a name or expression, with each letter pronounced separately.

truncation

An abbreviation formed from the first part of a word.

Numerals

cardinal

A value indicating quantity.

ordinal

A value indicating rank or position.

ratio

An expression of the magnitude of two quantities relative to each other.

percentage

A value that defines proportionality to a whole, including percent sign if present.

phone

A phone number, including area code and international dialing code.

isbn

An International Standard Book Number.

currency

A currency value, including the denomination indicator.

postal-code

A postal or zip code.

result

A single value that includes two or more sets of numbers separated by spaces.

fraction

A single fraction with a numerator and denominator, and without a preceding whole number.

mixed

A whole number followed by a fraction with a numerator and denominator.

decimal

A single value where the integral number is separated from the fractional value by a decimal point or comma.

roman

A single value expressed using roman numerals.

weight

A single weight value, including units.

measure

A single value of measure, including units.

coordinate

A value expressing the location of a point in n-dimensional space, including degree, minute and other indicators if specified.

Value Groups

range

A numeric, spatial or temporal range of values, as expressed by the start and end values.

result

The outcome of an event, such as a sporting match or vote.

Proper Nouns

place

The name of a city, state, province, country or other geographic or political entity. .

nationality

The name of a nationality, national group, or other designator of association. .

organization

The name of an organization, corporation, or other business entity. .

taxonomy

The name of a species, genus or other taxonomic classification. .

product

The brand name of a product, medication, etc. .

event

The name of a religious, sporting, or other event, holiday or festival. .

award

The name of an award or prize.

Personal names

personal-name

A proper name identifying a person, often composed of a given name and a family name.

Inherits from: http://xmlns.com/foaf/spec/#term_name

given-name

The part of a personal name that distinguishes the person from a group of persons sharing the same surname. Also known as 'forename' or 'first name'. .

Equivalent to: http://xmlns.com/foaf/spec/#term_givenName

surname

The part of a personal name that associates the person with a group of persons sharing the same surname. Also known as 'last name'.

See also: family-name

family-name

The part of a personal name that indicates the family to which the person belongs. Often used in conjunction with a honorific.

Inherits from: surname

See also: surname

Equivalent to: http://xmlns.com/foaf/spec/#term_familyName

name-title

The part of a personal name that signifies veneration, an official position or a professional or academic qualification. .

Equivalent to: http://xmlns.com/foaf/spec/#term_title

signature

The handwritten, digital or stamped representation of a person's name.

Words

word

A lexical word.

Inherits from: text

compound

A word created by the combination of two or more stems.

homograph

An instance of a set of words with the same spelling but different etymologies and often pronunciations (e.g. bass the instrument and bass the fish).

portmanteau

A word created from the merging of two other words (e.g. brunch from breakfast and lunch).

Word Parts

root

The primary lexical unit of a word.

stem

The word form common to all its inflected variants.

prefix

An affix attached to the front of a stem.

suffix

An affix attached to the end of a stem.

morpheme

The smallest linguistic unit of a word with semantic meaning.

phoneme

The smallest linguistic unit of a word that forms a sound.

grapheme

The smallest linguistic unit of a written word.

Artwork

illustration

An illustration.

photograph

A photograph.

decorative

Artwork serving only a decorative purpose, not contributing relevant information.

publisher-logo

The publisher's logo.

Inherits from: decorative

frontispiece

A frontmatter illustration.

Inherits from: illustration

References

reference

A reference to another information entity—a resource, fragment, concept, or notion.

resolving-reference

A reference which contains data intended to allow resolving the target entity.

Inherits from: reference

nonresolving-reference

A reference to a non-resolving target - a concept, notion or other imprecisely defined entity.

Inherits from: reference

noteref

A reference to a note, typically appearing a superscripted symbol in the main body of text

Inherits from: resolving-reference

See also: note

annoref

A reference to an annotation.

Inherits from: resolving-reference

See also: annotation

citation

A reference to bibliographic work, resolving to an entry in a directly or indirectly identified bibliography

Inherits from: resolving-reference

See also: bibliography and nonresolving-citation

nonresolving-citation

An imprecise reference to a bibliographic work.

Inherits from: nonresolving-reference

See also: citation

continuation

A reference to another segment of the current work that constitutes the logical continuation of the current segment.

Inherits from: resolving-reference

continuation-of

A reference to another segment of the current work that the current segment is a continuation of.

Inherits from: resolving-reference

Pagination

pagebreak

A (sometimes valued) separator denoting the position where a break between two contiguous pages occurs in a paginated media.

page-header

Material, separate from the main body of text, appearing at the top of a printed page.

page-footer

Material, separate from the main body of text, appearing at the bottom of a printed page.

recto

The front side of a page.

verso

The back side of a page.

Custom Content (Republishing)

image-placeholder

A placeholder for a reference to a non-existing or deliberately removed image.

Inherits from: nonresolving-reference

Priority

primary

Indicates the current element has primary importance (for example, may identify the primary table of contents for a work or a primary source).

secondary

Indicates the current element has secondary importance.

tertiary

Indicates the current element has tertiary importance.

11.2.3 Z39.98-2012 Resource Directory Vocabulary

This vocabulary defines a set of terms for use in profile and feature resource directories. Requirements on the application of this vocabulary in resource directory documents are defined in 10.2, *Resource directory conformance definition*.

The canonical URI of the Resource Directory Vocabulary is http://www.daisy.org/z3998/2012/vocab/resourcedirectory/.

Nature and Identity

reference

A human-readable description of the purpose and structure of the entity described by this directory.

token-identifier

A canonical string identifier for the entity described by this directory.

maintenance-agency

The identity of agency responsible for maintaining the entity described by this directory.

specification-compliance

A reference to a specification which the entity described by this directory conforms to.

version

A version identifier for the entity described by this directory.

current-version-uri

A URI resolving to the resource directory of the most recently published (current) version of the entity described by this directory.

version-history

A reference to list of URIs that contains all available versions of the entity described by this directory.

Instance Conformance

Resources related to testing whether instances satisfies the constraints expressed in this directory.

normative-schemata

The normative schema(ta) for validation of instances satisfying the constraints expressed in this directory.

normative-prose

Normative prose regarding the application of the entity described by this directory.

schema-documentation

Human-readable documentation of a schema.

prefix

A prefix functioning as a symbolical short form for a URI.

conformance

The conformance requirement of a resource: normative or informative.

format

The language in which a schema is expressed.

Instance Conformance (Z39.98-2012 Profile Specific)

Resources related to testing whether instances satisfies the constraints expressed in this directory

z3998-instance name

The name by which a profile or feature must be identified in Z39.98-AI document declarations.

z3998-feature

A Feature supported by the Profile.

z3998-rdfa-context

The canonical URI of the RDFa initial context document.

Auxiliary Resources

examples

Examples of instances satisfying the constraints expressed in this directory.

informative-schema-alt

An informative schema, alternative to normative schemas, for validation of instances satisfying the constraints expressed in this directory.

informative-schema-add

An informative schema, for use in addition to normative schemas.

module

A subcomponent of a schema.

default-css

A cascading stylesheet suitable for use in instances satisfying the constraints expressed in this directory.

supporting-software

Software that supports the entity described by this directory or instances satisfying the constraints expressed herein.

archive

A downloadable archive of documents and resources referenced from this directory.

11.3 Associating vocabularies with Z39.98-AI documents

An initial context document, as defined in section 9 of [RDFa], is an [XHTML+RDFa] document that defines the default RDF vocabulary for a Z39.98-AI profile (enabling the use of unprefixed CURIEs) as well as any additional vocabulary prefixes directly supported. Each Z39.98-AI profile must specify the URI of its initial context document in its resource directory using the term `z3998-rdfa-context` from the Z39.98-2012 Instance Metadata Vocabulary (see 6.3.3, *Profile resource directory*).

Z39.98-AI profile documents must identify their initial context document in the `resource` attribute of a `meta` element specifying the relationship `z3998:rdfa-context` in its `rel` attribute.

Example 16: Declaring an initial context document

```
…
<head>
  …
  <meta rel="z3998:rdfa-context"
resource="http://www.daisy.org/z3998/2012/vocab/context/default/" />
  …
</head>
…
```

When processing a document, the default vocabulary and any prefixes defined in the initial context document must be used to dereference any CURIE values found in RDFa-compliant

attributes. If, however, a child element defines its own metadata container, such as the `article` element, that element may define a localized context in their child metadata container element by the same means. When a localized context is defined, processors must ignore the context defined by the host document.

Only CURIEs from the default vocabulary defined in the initial context document may be referenced in their unprefixed form. The initial context document may also define CURIE prefix bindings, and these must be used to reference the associated vocabularies.

A vocabulary that has not been defined in the initial context document may still be referenced and used to attach metadata to a document. In this case, a prefix definition must be added to the document using the RDFa `prefix` attribute on the root `document` element. Prefixes are declared in whitespace separated pairs of the form:

```
NCName: URI
```

The `prefix` attribute must not be used to redeclare prefixes or URIs that have already been defined in the initial context document. The `xmlns` attribute must not be used to declare RDFa vocabulary prefixes.

All unprefixed CURIEs must resolve to a term in the default vocabulary in order for a document to be valid. It is not valid to use a CURIE prefix or reference terms from a vocabulary that has not been defined in the initial context document or `prefix` attribute.

A Z39.98-AI profile must not impose restrictions on the occurrence of references to vocabularies, except as relates to the initial context document.

11.4 Format of RDF vocabularies

This specification does not impose any restrictions on the format of RDF vocabularies associated with Z39.98-AI documents.

When developing vocabularies intended for use within the Z39.98-AI Framework, RDFa in XHTML, as defined by [XHTML+RDFa], is recommended to be the principal publication form. Implementers should be aware that vocabularies expressed in for example RDF Schema [RDFS] or Notation3 [N3] may also occur.

11.5 Changes to RDF vocabularies

Vocabularies are evolving documents, just like the profiles that reference them. The terms they contain may be modified or added to as required to augment the needs of profile and document creators. Terms should not be removed from vocabularies, however. If a term is no longer needed or has been replaced by another, its use should be deprecated.

Vocabularies are not required to identify their current version in their canonical URI, but should include a revision history. Processing agent developers consequently need to be aware that

changes may occur independently of profiles and take measures to ensure that inconsistencies do not arise.

12 Processing agents

12.1 Introduction

Processing agents are the applications used to interact with a Z39.98-AI document during its lifecycle, from its creation through to its provision to end users. Primary examples include programs that facilitate the markup of documents (XML editors and word processing applications), programs that check document integrity (validators), and programs that process documents to create final output formats (transformation pipelines). Note that the definition of processing agent as used in this specification does not include applications used for consumption by end users, as this specification does not cover the user-oriented formats that may result from the transformation of a Z39.98-AI document.

Processing agents are not expected to be able to handle any Z39.98-AI document that they encounter. A processing agent declares support for profiles individually, and must abort processing whenever it encounters a document conforming to a profile that it does not support, and by default also when encountering a document conforming to a supported profile but with an unsupported version number (either newer or older). Although this restriction is severe, it ensures that documents are not rendered in a way that contradicts their intended use, and that any resulting outputs are not prone to bugs resulting from the passive application of new or out-dated rules. This draconian abort principle is primarily motivated by the anticipated use of the Z39.98-AI framework in fully automated provisioning contexts where errors must not pass through silently, and where the integrity of the output cannot be manually controlled.

A processing agent's support for features is independent of its support for profiles, and by default the same draconian abort principle applies when encountering an unrecognized or unsupported feature. As described in 12.3, *Initialization*, however, features may define fallback methods that allow processing agents to adapt their behavior in a controlled manner and continue processing the document in spite of the fact that a given feature is not supported.

All processing agents should provide complete profile and feature support information to users in order to reduce the likelihood that they will be used to process incompatible documents and to ensure that they conform to user expectations.

12.2 Processing agent conformance definition

A conformant processing agent must meet all of the following criteria:

- It must adhere to all conformance requirements on non-validating XML processors as defined in [XML].
- It must adhere to all conformance requirements on processors as defined in [XMLNAMES].

- It must adhere to all conformance requirements on applications as defined in [XMLID].
- It must adhere to all conformance requirements on applications as defined in [XMLBase].
- It must detect and handle document profile and feature references as specified in 12.3, *Initialization*.
- It must comply to the requirements for processing RDF vocabulary terms specified in 12.4, *Processing of vocabulary terms*.
- It must be able to recognize by their `mimetype`, and unbundle, Z39.98-AI document sets packaged according to the container file format detailed in 9.2, *Format*.
- If claiming to support a given profile or feature, it must adhere to all behavioral rules defined by the given profile or feature through the corresponding resource directory and in the associated definitions.

In addition, a conformant processing agent should be a conformant application as defined in XML Inclusions [XInclude], including support for Namespace Fixup, Base URI Fixup and Language Fixup of the result infoset. If it is not a conformant application as defined in XML Inclusions, it must, if set to process a document that contains elements in the XInclude namespace, issue a notification and abort processing.

A conforming processing agent is not required to be validating (i.e., support validating a given Z39.98-AI document against the schema resource(s) defined as normative in the resource directory of a given profile). A conforming processing agent may abort processing if set to process a document that is known to be invalid.

12.3 Initialization

Processing agents must execute the following ordered sequence of steps during the initial stage of processing a Z39.98-AI document:

1. The processing agent attempts to retrieve the profile identity URI, available at the document location defined in 8.3, *Referencing profiles and features*.
2. a. If retrieval of the profile identity URI fails, then the processing agent must issue a notification and abort processing.

   b. If retrieval is successful, the processing agent analyzes the identity URI in order to ascertain whether the document adheres to a profile that the processing agent supports:

   - If the profile identity URI excluding the version segment represents a profile that is not supported by the processing agent, then it must issue a notification and abort processing.
   - If the version segment represents a profile version that is not supported by the processing agent, then it must issue a notification and abort processing, unless explicitly instructed by the client to continue processing under this particular circumstance.

3. The processing agent retrieves any feature identity URIs, available at the document location defined in 8.3, Referencing profiles and features.

   The processing agent analyses each identity URI in turn in order to ascertain whether the feature represented by the URI is either *supported*, *recognized but not supported*, or *not recognized*:

   Not recognized

   > If the feature identity URI excluding the version segment represents a feature that is not recognized by the processing agent, then it must issue a notification and abort processing.

   Recognized but not supported

   > If the feature identity URI represents a feature that is recognized but not supported by processing agent, then it must issue a notification and abort processing, unless normative prose available through the feature's resource directory specifies a behavioral fallback to employ under this particular circumstance.

   Supported

   > If the feature identity URI excluding the version segment represents a feature that is supported by the processing agent, then it processes the version segment in the same way as for profile version segments.

4. If the processing agent reaches this step without having aborted processing, it has completed initialization successfully.

12.4 Processing of vocabulary terms

The following rules apply to processing agents with regard to the processing of RDF vocabulary terms:

- A processing agent must recognize the terms used to express the required document-level metadata (see 8.5.2, *Required document-level metadata*), and the terms used to express the profile to which the document conforms (see 8.3, *Referencing profiles and features*).
- If a processing agent claims support for a profile and the profile defines a default RDF vocabulary, then the processing agent should natively recognize all terms in that vocabulary.
- A processing agent may associate specialized behaviors with none, some, or all of the RDF vocabulary terms that it natively recognizes.
- A processing agent should support the dereferencing of unrecognized CURIEs occurring in Z39.98-AI documents. If a processing agent successfully dereferences an unrecognized CURIE, it may (depending on the nature of and RDF data available in the referenced vocabulary property) employ adaptive behaviors in response to the change to the document structural semantics that results from the CURIE dereferencing.

- If the attempt of a processing agent to dereference an unrecognized CURIE fails, then the processing agent should issue a notification to the client, and then must continue processing the document, basing its behavior on the carrying XML element alone.
- If a processing agent does not support the dereferencing of unrecognized CURIEs, then the processing agent must upon encountering an unrecognized CURIE continue processing the document, basing its behavior on the carrying XML element alone.
- A processing agent should recognize the implicit metadata association mechanism defined for micro-documents in 8.5.4, *RDFa metadata associations*.
- When a vocabulary term is used for semantic inflection and processing agent behavior associated with that term contradicts behavior associated with the carrying XML element, then the behavior associated with the element must take precedence.

Appendix A:
(normative)
Profile, feature, and vocabulary catalogs

Table of Contents

All catalogs listed in this appendix are maintained and hosted by the DAISY Consortium under the guidance of its Technical Advisory Committee. For more information about the Committee, including its current membership and how to participate, please refer to the DAISY Consortium Web site.

A.1 Profile catalog

The catalog of profiles is located at http://www.daisy.org/z3998/2012/auth/profiles/.

A.2 Feature catalog

The catalog of features is located at http://www.daisy.org/z3998/2012/auth/features/.

A.3 Vocabulary catalog

The catalog of RDF vocabularies is located at http://www.daisy.org/z3998/2012/vocab/.

This catalog normatively defines the URIs of the core vocabularies that are predefined by this specification (see 11.2, *Vocabularies defined by this specification*, and may be extended to reference additional vocabularies.

Appendix B:
(normative)
Schema languages

The grammars for profiles, modules, and features must be expressed using one or more of the following schema languages:

- RelaxNG [RelaxNG]
- NVDL [NVDL]
- ISO Schematron [ISOSchematron]
- W3C XML Schema 1.0 [XML Schema Part1] [XML Schema Part2]

Appendix C:
(normative)
Media type registration

Table of Contents

C.1 Z39.98-AI XML Documents

Type name
application

Subtype name
z3998-auth+xml

Required parameters
None

Optional parameters
charset

The same semantics as the `charset` parameter of application/xml, as specified in [RFC3023].

Encoding considerations
As Z39.98-AI documents are XML documents, the same encoding considerations apply as for application/xml, as specified in [RFC3023].

Security considerations
None

Interoperability considerations
Z39.98-AI documents are structured to conform to unique profiles, which means that only well-formedness must be expected. Rules for processing conformant and non-conformant content are provided within this document.

Published specification
This media type registration is extracted from Appendix D of the Z39.98 Authoring and Interchange Framework for Adaptive XML Publishing Specification.

Applications which use this media type
There are no applications that currently recognize this media type. This new type is being registered to enable operation within XML editors, word processors, web browsers, and in text-to-speech playback devices.

Additional information

Magic number(s):

No single initial set of bytes uniquely identifies Z39.98-AI documents. See [RFC3023] for common considerations for XML documents.

File extension(s):

The extension .xml is primarily used to identify Z39.98-AI documents, but not exclusively.

Macintosh file type code(s):

TEXT

Intended usage

COMMON

Restrictions on usage

None

Person to contact for further information

Name

Markus Gylling

Email

mgylling@daisy.org

Author/Change controller

The Z39.98-AI specification is maintained by the Technical Advisory Committee of the DAISY Consortium (the NISO-appointed Maintenance Agency).

C.2 Z39.98-AI container format

Type name

application

Subtype name

z3998-auth+zip

Required parameters

None

Optional parameters

None

Encoding considerations

Z39.98-AI container files are binary files in ZIP format.

Security considerations

The file size and validity of container file data should always be rigorously verified by processors.

Interoperability considerations

None

Published specification

This media type registration is extracted from Appendix C of the Z39.98 Authoring and Interchange Framework for Adaptive XML Publishing Specification.

Applications which use this media type

There are no applications that currently recognize this media type. This new type is being registered to enable operation within XML editors, word processors, validators, document transformation applications, and applications that perform content transfer and/or retrieval operations.

Additional information

Magic number(s):

0: `PK`, 30: `mimetype`, 38: `application/z3998-auth+zip`

File extension(s):

`.zai`

Macintosh file type code(s):

`ZIP`

Intended usage

COMMON

Restrictions on usage

None

Person to contact for further information

Name

Markus Gylling

Email

mgylling@daisy.org

Author/Change controller

The Z39.98-AI specification is maintained by the Technical Advisory Committee of the DAISY Consortium (the NISO-appointed Maintenance Agency).

Informative References

This specification makes reference to the following informative resources:

[CMoS]
The Chicago Manual of Style. 16th ed.
Chicago: The University of Chicago Press, 2010.
Online version available at: http://www.chicagomanualofstyle.org/home.html

[MathML]
Mathematical Markup Language (MathML) Version 3.0.
W3C (World Wide Web Consortium). 21 October 2010.
http://www.w3.org/TR/2010/REC-MathML3-20101021/

[N3]
W3C Team Submission: Notation3 (N3): A readable RDF syntax.
W3C (World Wide Web Consortium). 28 March 2011.
http://www.w3.org/TeamSubmission/2011/SUBM-n3-20110328/

[OPF]
Open Packaging Format (OPF) 2.0 v1.0.
IDPF (International Digital Publishing Forum). 11 September 2007.
http://www.idpf.org/2007/opf/OPF_2.0_final_spec.html

[RDFS]
RDF Vocabulary Description Language 1.0: RDF Schema.
W3C (World Wide Web Consortium). 10 February 2004.
http://www.w3.org/TR/2004/REC-rdf-schema-20040210/

[ROLE]
W3C Working Draft: Role Attribute 1.0.
W3C (World Wide Web Consortium). 5 December 2010.
http://www.w3.org/WAI/PF/role-attribute/

[XProc]
XProc: An XML Pipeline Language.
W3C (World Wide Web Consortium). 9 March 2010.
http://www.w3.org/TR/2010/REC-xproc-20100511/

[Z39.86-2005]
ANSI/NISO Z39.86-2005: Specifications for the Digital Talking Book.
NISO (National Information Standards Organization). 21 April 2005.
http://www.niso.org/standards/z39-86-2005/