



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This standard may be revised or withdrawn at any time.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

Copyright © 2012 by the National Information Standards Organization

All rights reserved under International and Pan-American Copyright Conventions. For noncommercial purposes only, this publication may be reproduced or transmitted in any form or by any means without prior permission in writing from the publisher, provided it is reproduced accurately, the source of the material is identified, and the NISO copyright status is acknowledged. For permission to photocopy or use material electronically from this publication, please access www.copyright.com or contact the Copyright Clearance Center, Inc. (CCC) at 222 Rosewood Drive, Danvers, MA 01923, 978-750-8400. All inquiries regarding translations into other languages or commercial reproduction or distribution should be addressed to NISO, One North Charles Street, Suite 1905, Baltimore, MD 21201.

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

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

Table of Contents

Foreword.....	vii
1 Purpose and Scope.....	1
1.1 Audience	1
1.2 Design goals.....	1
1.3 Overview.....	2
1.4 Relationship to other specifications	5
1.4.1 ANSI/NISO Z39.86-2005	5
1.4.2 XML.....	5
1.4.3 Unicode.....	5
1.5 Conformance statements.....	5
2 Normative References.....	5
3 Terms and Definitions	8
4 Abstract Document Model.....	10
4.1 Introduction.....	10
4.2 Fundamentals	12
4.2.1 Document foundation.....	12
4.2.2 Document structure layers	12
4.2.3 Document attribute collection.....	13
4.2.4 Constraints	14
4.2.5 Classes.....	15
4.3 Layer and Collection Definitions.....	15
4.3.1 Introduction.....	15
4.3.2 Document foundation.....	16
4.3.3 Section layer.....	16
4.3.3.1 Description	16
4.3.3.2 Implementation	17
4.3.4 Block layer	17
4.3.4.1 Description.....	17
4.3.4.2 Implementation	17
4.3.5 Phrase layer.....	18
4.3.5.1 Description.....	18

4.3.5.2 Implementation	18
4.3.6 Text layer	18
4.3.6.1 Description	18
4.3.6.2 Implementation	18
4.3.7 Metadata Attributes collection	19
4.3.7.1 Description	19
4.3.7.2 Implementation	19
4.3.8 Global Attributes Collection	20
4.3.8.1 Description	20
4.3.8.2 Implementation	20
5 Modules	20
5.1 Introduction	20
5.2 Module and component definitions	20
5.3 Components	21
5.3.1 Outline	21
5.3.2 Semantic definition	21
5.3.3 Default usage context	22
5.3.4 Default content model	22
5.3.5 Default attribute model	23
5.3.6 Definition of alterability	23
5.3.7 Definition of optionality	23
5.4 Expression	24
5.5 Activation	24
5.6 Core modules	25
5.6.1 Core namespace URI	25
5.6.2 Core modules URI	25
6 Profiles	26
6.1 Introduction	26
6.2 Creation	26
6.3 Profile conformance definition	28
6.3.1 Profile identity URI	28
6.3.2 Profile markup model definition	28

6.3.3	Profile resource directory.....	29
6.3.4	RDFa initial context.....	29
7	Features	30
7.1	Introduction.....	30
7.2	Feature conformance definition	30
7.2.1	Feature identity URI	30
7.2.2	Feature markup model definition.....	31
7.2.3	Feature resource directory.....	31
8	Documents	32
8.1	Introduction.....	32
8.2	Document conformance definition	32
8.3	Referencing profiles and features	33
8.4	Referencing RDF vocabularies.....	34
8.5	Metadata.....	35
8.5.1	Introduction.....	35
8.5.2	Required document-level metadata.....	36
8.5.3	Document-level metadata resources	37
8.5.4	RDFa metadata associations	38
8.5.5	Inline metadata.....	39
9	Container	40
9.1	Introduction	40
9.2	Format.....	40
9.3	File extension.....	42
9.4	Media types.....	42
10	Resource directories	42
10.1	Introduction.....	42
10.2	Resource directory conformance definition.....	43
11	RDF vocabularies	43
11.1	Introduction.....	43
11.2	Vocabularies defined by this specification	44
11.2.1	Z39.98-2012 Instance Metadata Vocabulary.....	44
11.2.2	Z39.98-2012 Structural Semantics Vocabulary.....	46

11.2.3 Z39.98-2012 Resource Directory Vocabulary	69
11.3 Associating vocabularies with Z39.98-AI documents	72
11.4 Format of RDF vocabularies.....	73
11.5 Changes to RDF vocabularies.....	73
12 Processing agents	74
12.1 Introduction.....	74
12.2 Processing agent conformance definition	74
12.3 Initialization	75
12.4 Processing of vocabulary terms	76
Appendix A: (normative) Profile, feature, and vocabulary catalogs	78
A.1 Profile catalog	78
A.2 Feature catalog	78
A.3 Vocabulary catalog	78
Appendix B: (normative) Schema languages	79
Appendix C: (normative) Media type registration	80
C.1 Z39.98-AI XML Documents	80
C.2 Z39.98-AI container format	81
Informative References	83
 List of Examples	
Example 1: Document foundation as an XML document	12
Example 2: Document structure layers depicted as XML	13
Example 3: Example of metadata attributes	14
Example 4: Example of global attributes.....	14
Example 5: Modification of an RNG module during activation.....	24
Example 6: Referencing a profile and features from a Z39.98-AI document	33
Example 7: Referencing an RDFa initial context document	34
Example 8: Referencing a term from the default vocabulary	34
Example 9: Referencing a term from a prefixed vocabulary	34
Example 10: Referencing an RDFa vocabulary.....	35
Example 11: Invalid vocabulary prefix declaration using the xmlns attribute	35
Example 12: Z39.98-AI document metadata	38
Example 13: Inline metadata	40
Example 14: Z39.98-AI container file structure	41
Example 15: container.xml file mark up.....	42
Example 16: Declaring an initial context document.....	72

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 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

Judith Gibbs

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

ANSI/NISO Z39.98-2012

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](#).

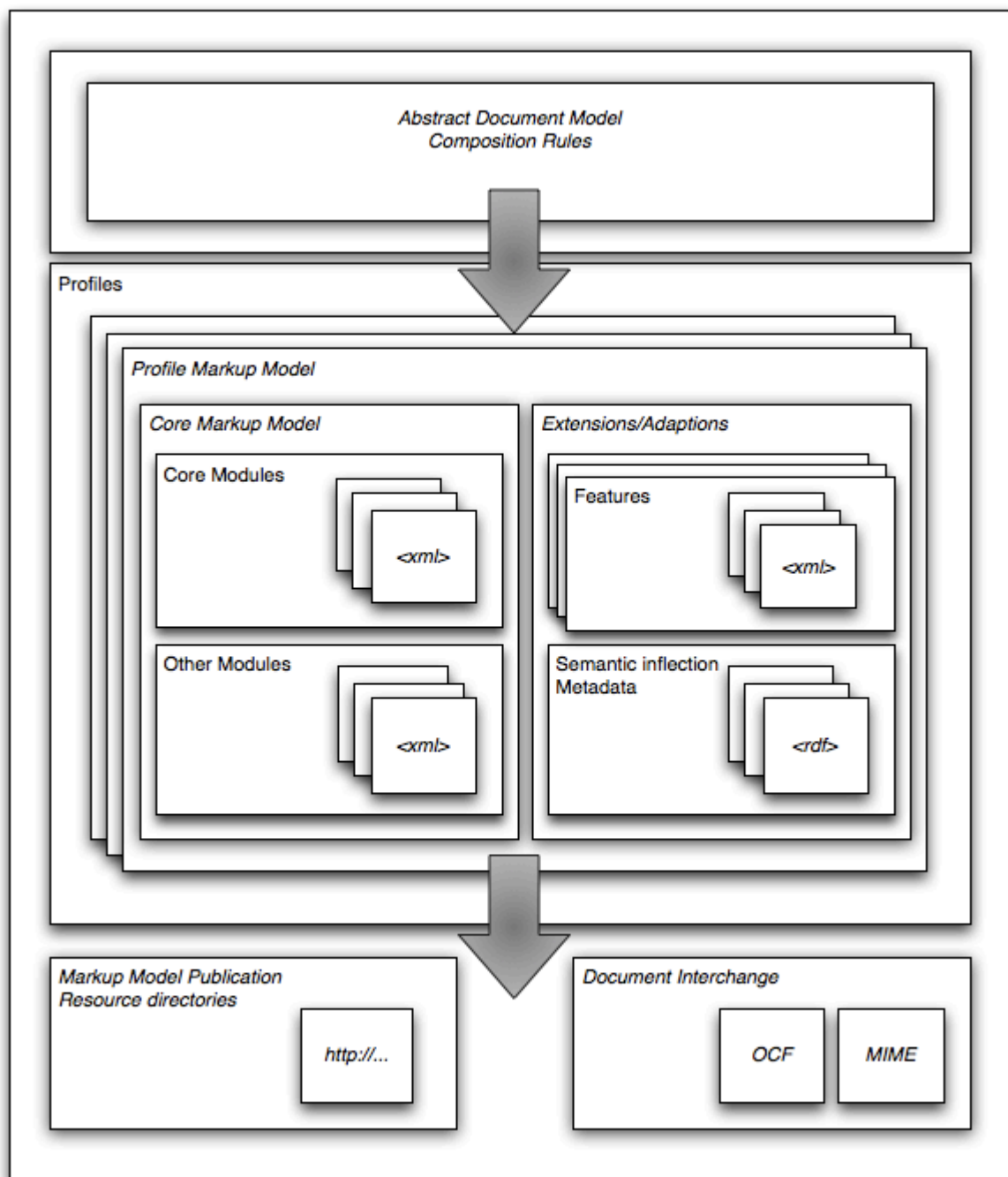


Figure 1: Overview of the Z39.98-AI profile creation process

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](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](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](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.

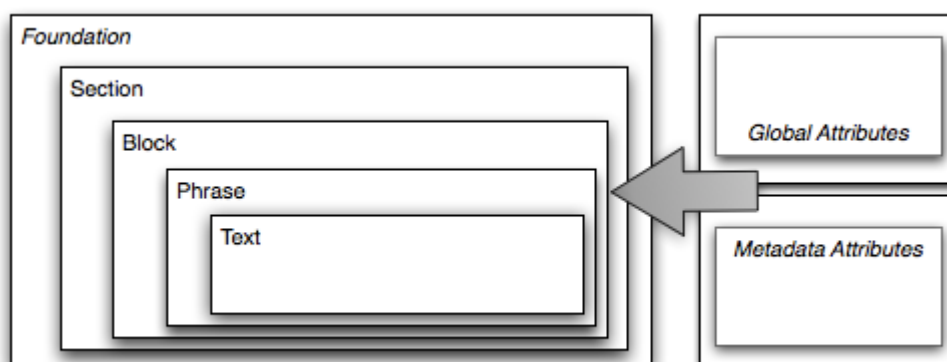


Figure 2: The document foundation and four layers of the Abstract Document Model

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>
</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