

The OpenURL Framework for Context-Sensitive Services

Abstract: The OpenURL Framework Standard defines an architecture for creating OpenURL Framework Applications. An OpenURL Framework Application is a networked service environment, in which packages of information are transported over a network. These packages have a description of a referenced resource at their core, and they are transported with the intent of obtaining context-sensitive services pertaining to the referenced resource. To enable the recipients of these packages to deliver such context-sensitive services, each package describes the referenced resource itself, the network context in which the resource is referenced, and the context in which the service request takes place.

This Standard specifies how to construct these packages as Representations of abstract information constructs called ContextObjects. To this end, the OpenURL Framework Standard defines the following core components: Character Encoding, Serialization, Constraint Language, ContextObject Format, Metadata Format, and Namespace. In addition, this Standard defines Transport, a core component that enables communities to specify how to transport ContextObject Representations. Finally, this Standard specifies how a community can deploy a new OpenURL Framework Application by defining a new Community Profile, the last core component.

This Standard defines the OpenURL Framework Registry and the rules that govern the usage of this Registry. The OpenURL Framework Registry contains all instances of all core components created by communities that have deployed OpenURL Framework Applications. This Standard defines and registers the initial content of the OpenURL Framework Registry, thereby deploying two distinct OpenURL Framework Applications.

An American National Standard
Developed by the
National Information Standards Organization

Approved: April 15, 2005
by the
American National Standards Institute
Reaffirmed May 13, 2010

Published by the National Information Standards Organization
Baltimore, Maryland, U.S.A.

ANSI/NISO Z39.88-2004 (R2010)

About NISO Standards

NISO Standards are developed by the Working Groups and Standing Committees of the National Information Standards Organization. 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:
<http://www.niso.org>

Published by

NISO
3600 Clipper Mill Road
Suite 302
Baltimore, MD 21211
www.niso.org

Copyright © 2010 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. All inquiries regarding translations into other languages or commercial reproduction or distribution should be addressed to: NISO 3600 Clipper Mill Road, Suite 302, Baltimore, MD 21211.

ISSN: 1041-5653 National Information Standards series

ISBN: 978-1-937522-38-4

Contents

Foreword	v
1 Purpose and Scope	1
2 Referenced Standards	2
3 Notational Conventions	4
4 Definitions	5
Part 1: ContextObjects and Transports	9
5 ContextObject, Entity, and Descriptor	11
5.1 ContextObject and Entity	11
5.2 Descriptor	12
5.2.1 Identifier	12
5.2.2 By-Value Metadata	13
5.2.3 By-Reference Metadata	13
5.2.4 Private Data	13
5.3 Constraints	14
6 Registry	15
6.1 Registry Entries	15
6.2 Registry Identifiers	17
6.3 Using the Registry	17
7 Formats	18
7.1 Serializations [Registry]	19
7.2 Constraint Languages [Registry]	20
7.3 Constraint Definitions	20
8 Representing ContextObjects	21
8.1 Character Encodings [Registry]	21
8.2 ContextObject Formats [Registry]	22
9 Representing Entities	24
9.1 Namespaces [Registry]	24
9.2 Metadata Formats [Registry]	25
10 Transporting ContextObject Representations: Transports [Registry]	27
11 Defining Applications: Community Profiles [Registry]	28
Part 2: The KEV ContextObject Format	31
12 The KEV ContextObject Format	33
12.1 The KEV Serialization	33
12.2 The Z39.88-2004 Matrix Constraint Language	34
12.3 Constraint Definitions in the KEV ContextObject Format	35
12.3.1 Z39.88-2004 Matrix Constraint Definition for the KEV ContextObject Format	35
12.3.2 Z39.88-2004 Matrix Constraint Definitions for KEV Metadata Formats	36

13 KEV ContextObject Representations	37
13.1 Cardinality Constraints on the KEV ContextObject Format.....	38
13.2 Keys in the KEV ContextObject Format	38
13.2.1 Keys for Entity Descriptors.....	39
13.2.2 Keys for By-Value Metadata Descriptors	40
13.2.3 Keys for By-Reference Metadata Descriptors	40
13.2.4 Keys for Administrative Data.....	40
13.3 Character Encoding in the KEV ContextObject Format	40
13.4 URL-Encoding in the KEV ContextObject Format.....	41
14 Entity Descriptors in the KEV ContextObject Format.....	41
14.1 Identifier Descriptors.....	41
14.2 By-Value and By-Reference Metadata Descriptors	42
14.2.1 Rules Guiding By-Value and By-Reference Metadata Descriptors	42
14.2.2 By-Value Metadata Descriptors	43
14.2.3 By-Reference Metadata Descriptors.....	43
14.3 Private Data Descriptors.....	44
14.4 Example of a KEV ContextObject Representation.....	45
15 KEV-Based Community Profiles	46
Part 3: The XML ContextObject Format	47
16 The XML ContextObject Format.....	49
16.1 The XML Serialization	49
16.2 XML Schema as a Constraint Language.....	49
16.3 Constraint Definitions in the XML ContextObject Format.....	49
16.3.1 XML Schema Constraint Definition for the XML ContextObject Format.....	50
16.3.2 XML Schema Constraint Definitions for XML Metadata Formats	54
17 XML ContextObject Representations	62
17.1 Cardinality Constraints on the XML ContextObject Format	63
17.2 Entity and Descriptor Elements in the XML ContextObject Format	64
17.3 Administrative Elements and Attributes in the XML ContextObject Format.....	66
17.4 Character Encoding in the XML ContextObject Format	66
18 Entity Descriptors in the XML ContextObject Format	66
18.1 Identifier Descriptors.....	67
18.2 By-Value and By-Reference Metadata Descriptors	67
18.2.1 Rules Guiding By-Value and By-Reference Metadata Descriptors	67
18.2.2 By-Value Metadata Descriptors	68
18.2.3 By-Reference Metadata Descriptors.....	69
18.3 Private Data Descriptors.....	69
18.4 Example of an XML ContextObject Representation.....	70
19 XML-Based Community Profiles	71
Part 4: OpenURL Transports	73
20 By-Reference OpenURL Transports	75
20.1 OpenURL Keys in By-Reference OpenURL Transports	75
20.2 By-Reference OpenURL Transports using HTTP(S) GET	76
20.3 By-Reference OpenURL Transports using HTTP(S) POST	77

21 By-Value OpenURL Transports	77
21.1 OpenURL Keys in By-Value OpenURL Transports	78
21.2 By-Value OpenURL Transports using HTTP(S) GET	79
21.3 By-Value OpenURL Transports using HTTP(S) POST	80
22 Inline OpenURL Transports	81
22.1 OpenURL Keys in Inline OpenURL Transports	82
22.2 Inline OpenURL Transports using HTTP(S) GET	83
22.3 Inline OpenURL Transports using HTTP(S) POST	84
Appendix A Responsibilities of the Maintenance Agency for the OpenURL Framework Standard (informative)	87
Appendix B Specification of the Z39.88-2004 Matrix Constraint Language (normative)	89
B.1 The Z39.88-2004 Matrix Constraint Language	89
B.2 Constraint Definitions in the KEV ContextObject Format	90
Appendix C The Level 1 San Antonio Community Profile (informative)	95
C.1 History	95
C.2 Maintenance of SAP1	95
C.3 Introduction to SAP1	95
C.4 Purpose and Scope	96
C.5 Registry Entries in SAP1	97
Appendix D The Level 2 San Antonio Community Profile (informative)	99
D.1 History	99
D.2 Maintenance of SAP2	99
D.3 Introduction to SAP2	99
D.4 Purpose and Scope	99
D.5 Registry Entries in SAP2	101
Appendix E Implementation Guidelines for the OpenURL Transports (informative)	103
E.1 Length of HTTP(S) GET URIs	103
E.2 URL-Encoding and URL-Decoding	103
E.3 Parsing of HTTP(S) Query Strings	103

Figures

Figure 1: Core Components of the OpenURL Framework	16
--	----

Tables

Table 1: Fundamental ContextObject Constraints	14
Table 2: Core Components and their Registry Identifiers	17
Table 3: Registry Identifiers for Serializations	19
Table 4: Registry Identifiers for Constraint Languages	20
Table 5: Registry Identifiers for Character Encodings	22
Table 6: Registry Identifiers for ContextObject Formats	23
Table 7: Registry Identifiers for Namespaces	25
Table 8: Registry Identifiers for Registered Metadata Formats	26
Table 9: Registry Identifiers for Transports	27
Table 10: Registry Identifiers for Community Profiles	29
Table 11: SAP1 Community Profile, Excerpt	30

Table 12: Structure of the Z39.88-2004 Matrix	34
Table 13: Z39.88-2004 Matrix Constraint Definition of KEV ContextObject Format, Excerpt	36
Table 14: Z39.88-2004 Matrix Constraint Definition of KEV Metadata Format for “book”, Excerpt.....	37
Table 15: KEV ContextObject Format – Cardinality Constraints	38
Table 16: KEV ContextObject Format – Keys for Entity Descriptors	39
Table 17: KEV ContextObject Format – Administration Key Prefix and Suffixes	40
Table 18: XML Schema Constraint Definition of XML ContextObject Format	50
Table 19: XML Schema Constraint Definition of XML Metadata Format for “journal”	55
Table 20: XML ContextObject Format – Cardinality Constraints	64
Table 21: XML ContextObject Format – Entities and Descriptors	65
Table 22: XML ContextObject Format – Administrative Information	66
Table 23: Structure of the Z39.88-2004 Matrix	89
Table 24: XHTML Template for Z39.88-2004 Matrix	91
Table 25: Use of ContextObject Entities in the Scholarly-Information Community.....	96
Table 26: SAP1 Registered Elements	97
Table 27: Use of ContextObject Entities in the Scholarly-Information Community.....	100
Table 28: SAP2 Registered Elements	101

Examples

Example 1: Examples of Entities	12
Example 2: Identifiers for a Referent, Requester, and Resolver	12
Example 3: By-Value Metadata for a Referent	13
Example 4: By-Reference Metadata for a Requester	13
Example 5: Private Data for a Referent	14
Example 6: A Registry Entry	16
Example 7: Identification of a Character Encoding	22
Example 8: Identification of a ContextObject Format	24
Example 9: Identification of Entities using Identifiers from Namespaces	25
Example 10: Identification of Unregistered Metadata Formats.....	27
Example 11: KEV ContextObject Representation	38
Example 12: Identifier Descriptors in a KEV ContextObject Representation	41
Example 13: By-Value Metadata Descriptor in a KEV ContextObject Representation	43
Example 14: By-Reference Metadata Descriptor as a Property List	43
Example 15: Private Data Descriptor in a KEV ContextObject Representation	44
Example 16: KEV ContextObject Representation	45
Example 17: XML ContextObject Representation	63
Example 18: Identifier Descriptors in an XML ContextObject Representation	67
Example 19: Referent with a By-Value Metadata Descriptor.....	68
Example 20: Requester with a By-Reference Metadata Descriptor	69
Example 21: ReferringEntity with a Private Data Descriptor	69
Example 22: XML ContextObject Representation	70
Example 23: By-Reference OpenURL Transport using HTTP GET	76
Example 24: By-Reference OpenURL Transport using HTTP POST	77
Example 25: By-Value OpenURL Transport using HTTP GET	79
Example 26: By-Value OpenURL Transport using HTTP POST	80
Example 27: Inline OpenURL Transport using HTTP GET	83
Example 28: An HTML Form (POST Method) to generate an Inline OpenURL Transport	85
Example 29: Inline OpenURL Transport using HTTP POST	85

Foreword

(This foreword is not part of *The OpenURL Framework for Context Sensitive Services*, ANSI/NISO Z39.88-2004 (R2010). It is included for information only.)

History

As the World Wide Web began its explosive growth in the early 1990s, the scholarly-information community made available digital scholarly materials, consisting of metadata and full-text content. As this body of materials grew, it became increasingly difficult to provide adequate links between related information assets, distributed across many collections and controlled by different custodians. By 1999, the scholarly-information community had embarked on several linking efforts, surveyed in Van de Sompel and Hochstenbach [R1].

In 1999, NISO started a series of invitational workshops to explore issues in the area of reference linking. Representatives from the library, publishing, and information services communities identified the appropriate-copy problem as a major issue, because its solution was expected to solve other link-resolution problems. The appropriate-copy problem arises when multiple copies of a resource exist, and each copy is governed by a different access policy. A specific user should be directed to a copy of the resource that is governed by an access policy compatible with that user's access privileges. None of the proposed linking architectures could accomplish this.

A series of collaborations by Herbert Van de Sompel (Ghent University, Los Alamos National Laboratory, and Cornell University), Patrick Hochstenbach (Ghent University), and Oren Beit-Arie (Ex Libris) culminated in the solution of the appropriate-copy problem. Their solution also addressed related issues in the delivery of context-sensitive services for the web-based scholarly information environment. This collaboration resulted in:

- Development of the SFX linking server and the OpenURL architecture [R1] [R2] [R3].
- Publication of the OpenURL 0.1 specification that defines an HTTP GET syntax for transporting metadata and identifiers from an information service to a linking server. The transported metadata and identifiers describe a referenced item and some contextual information [R4].
- Publication of the OpenURL Framework, which provides a design for context-sensitive reference linking in the Web-based scholarly information environment [R5].

The scholarly-information community quickly embraced the OpenURL 0.1 specification. Publishers, vendors of abstracting and indexing databases, preprint systems, and CrossRef (a Registration Agency for Digital Object Identifiers or DOIs) introduced OpenURLs in their systems. Many libraries implemented OpenURL-conformant linking servers that provide their users with context-sensitive links. This quick adoption by so many constituents established OpenURL 0.1 as a de-facto standard.

In preparation for the NISO Standardization effort, Herbert Van de Sompel and Oren Beit-Arie studied OpenURLs in real environments and analyzed the information sent to OpenURL resolvers. Based on this analysis, they proposed the Bison-Futé model [R6], a generalization of OpenURL 0.1 based on the notion of a ContextObject. A ContextObject is an information construct that formalizes and generalizes the information packaged in OpenURL 0.1 requests:

OpenURL 0.1	ContextObject
The OpenURL 0.1 specification explicitly includes the referenced resource, the system that provides the OpenURL (sid), and the linking server that is the target of the OpenURL.	The ContextObject definition formalizes these resources into, respectively, the Referent, the Referrer, and the Resolver Entities.
OpenURL 0.1 usage showed that three more resources were regularly described in the Private Identifier (pid): the initiator of an OpenURL transport (the user clicking the OpenURL), the citing scholarly work, and the type of service requested (for example, "provide full-text").	The ContextObject definition formalizes these resources into, respectively, the Requester, the ReferringEntity, and the ServiceType Entities.
OpenURL 0.1 specifies one set of metadata keys to describe a referenced scholarly work by inline metadata.	The ContextObject definition generalizes this description method into the By-Value Metadata Descriptor of Entities. This Descriptor depends on Metadata Formats that are made available through registration in the OpenURL Framework Registry.
OpenURL 0.1 specifies namespaces for identifiers of referenced scholarly works, for the system that provides the OpenURL, and for the target of the OpenURL.	The ContextObject definition generalizes this description method into the Identifier Descriptor of Entities. This Descriptor depends on Namespaces for Identifiers that are made available through registration in the OpenURL Framework Registry.
OpenURL 0.1 allows for private data to describe a referenced scholarly work by a method that is specific to the provider of the OpenURL. To process this syntax, a linking server must enter into some prior agreement with the provider of the OpenURL.	The ContextObject definition generalizes this description method into the Private Data Descriptor of Entities.
OpenURL 0.1 usage showed that the Private Identifier (pid) of OpenURL 0.1 often contains a pointer to metadata describing the referenced scholarly work.	The ContextObject definition generalizes this description method into the By-Reference Metadata Descriptor of Entities. This Descriptor depends on Metadata Formats that are made available through registration in the OpenURL Framework Registry.

In 2001, NISO formed Committee AX to prepare this Standard. The Committee's charge [R7] was to develop an extensible mechanism for the representation and transportation of packages of metadata and identifiers that are useful in the delivery of context-sensitive services. The table above shows how the OpenURL 0.1 specification inspired the ContextObject concept of the Bison-Futé model. The Committee took this concept as the starting point for its work. To achieve extensibility, the Committee embedded the ContextObject concept in a general (and abstract) framework, called the OpenURL Framework.

This framework is defined in Part 1 and consists of the following core components: Namespaces for Identifiers, Character Encodings, Serializations, Constraint Languages, ContextObject Formats, Metadata Formats, Transports, and Community Profiles. To create an instantiation of the OpenURL

Framework for a particular application domain, a community must specify and register specific selections of these core components in the OpenURL Framework Registry.

Although it retains the name OpenURL in its title for historical reasons, the OpenURL Framework is neutral with respect to application domain. The Committee hopes that the ContextObject specified in this Standard will become a generic component for systems providing contextual services pertaining to resources that are referenced on networks.

References

- [R1] Van de Sompel H, Hochstenbach P. *Reference Linking in a Hybrid Library Environment, Part 1: Frameworks for Linking*. D-Lib Magazine, 1999. 5(4). doi:10.1045/april99-van_de_sompel-pt1 [online] [cited 4 November 2004] Available from World Wide Web: <http://www.dlib.org/dlib/april99/van_de_sompel/04van_de_sompel-pt1.html>
- [R2] Van de Sompel H, Hochstenbach P. *Reference Linking in a Hybrid Library Environment, Part 2: SFX, a Generic Linking Solution*. D-Lib Magazine, 1999. 5(4). doi:10.1045/april99-van_de_sompel-pt2 [online] [cited 4 November 2004] Available from World Wide Web: <http://www.dlib.org/dlib/april99/van_de_sompel/04van_de_sompel-pt2.html>
- [R3] Van de Sompel H, Hochstenbach P. *Reference Linking in a Hybrid Library Environment, Part 3: Generalizing the SFX solution in the "SFX@Ghent & SFX@LANL" experiment*. D-Lib Magazine, 1999. 5(10). doi:10.1045/october99-van_de_sompel [online] [cited 4 November 2004] Available from World Wide Web: <http://www.dlib.org/dlib/october99/van_de_sompel/10van_de_sompel.html>
- [R4] Van de Sompel H, Hochstenbach P, Beit-Arie O (editors). *OpenURL Syntax Description*. 2000. [online] [cited 4 November 2004] Available from World Wide Web: <<http://www.openurl.info/registry/docs/pdf/openurl-01.pdf>>
- [R5] Van de Sompel H, Beit-Arie O. *Open Linking in the Scholarly Information Environment Using the OpenURL Framework*. D-Lib Magazine, 2001. 7(3). doi:10.1045/march2001-vandesompel [online] [cited 4 November 2004] Available from World Wide Web: <<http://www.dlib.org/dlib/march01/vandesompel/03vandesompel.html>>
- [R6] Van de Sompel H, Beit-Arie O. *Generalizing the OpenURL Framework beyond References to Scholarly Works: the Bison-Futé model*. D-Lib Magazine, 2001. 7(7/8). doi:10.1045/july2001-vandesompel [online] [cited 4 November 2004] Available from World Wide Web: <<http://www.dlib.org/dlib/july01/vandesompel/07vandesompel.html>>
- [R7] National Information Standards Organization. *The OpenURL Framework for Context-Sensitive Services. Standards Committee AX. Charge*. [online] [cited 4 November 2004] Available from World Wide Web: <http://www.niso.org/committees/committee_ax.html>

Preamble

Wide adoption of any technology or process is often a result of its simplicity coupled with the effective meeting of a market need. OpenURL has been embraced and adopted by the scholarly-information community for these very reasons.

The early implementation of OpenURL was simple in concept: it used HTTP GET or POST to transfer information about an item (a journal article, for example) from an online service to a linking server. The specifications were simple. They described the protocol, the syntax, and how a referenced item is to be represented by using particular sets of data element names on a URL. This is best demonstrated by an example of an OpenURL:

`http://www.example.com/resolver?genre=article
&atitle=p27-p16 Chimera: A Superior Antiproliferative
&title=Molecular Theory
&aulast=McArthur
&aufirst=James
&date=2001
&volume=3
&issue=1
&spage=8
&epage=13`

Because of this simplicity, developers working for online service providers and developers of linking servers could quickly understand how OpenURL worked and develop their own products. The scholarly-information community responded by quickly adopting link resolvers as a basic component of its digital library infrastructure.

The Committee recognized the status of this early version of OpenURL as a de facto standard. The *OpenURL Framework Standard* (Z39.88-2004) refers to the early version as OpenURL 0.1, and the OpenURL 0.1 specifications are retained in the OpenURL Framework Registry <<http://www.openurl.info/registry/docs/pdf/openurl-01.pdf>>.

The simplicity of OpenURL 0.1 is due, in part, to the static nature of its syntax, the limited number of genres supported, the fixed sets of data elements, and the fixed transport protocol (HTTP). However, the fixed nature of OpenURL 0.1 not only limits its expansion within the scholarly-information community, it also prevents other communities from adopting OpenURL for similar needs. For example, OpenURL 0.1 cannot be extended to cover other genres of materials. It cannot even extend the data element sets for existing genres. The *OpenURL Framework Standard* is about providing such extensibility.

In developing this Standard, the Committee wanted to provide the needed extensibility while retaining the simplicity of the original OpenURL. To accomplish this, the fundamentals of an extensible framework had to be put in place. This document describes these fundamentals: the framework upon which OpenURL can be extended with new genres, new data elements, different character encodings (to support non-English use of the OpenURL), various network protocols, and data representations.

Part 1 (Sections 5 through 11) describes the ContextObject. The ContextObject is the information construct that describes an item that is the subject of a service request and the context within which the request is being made. While the term ContextObject may be new, the concept it represents is entirely compatible with OpenURL 0.1. Indeed, the original OpenURLs were precisely about a request to provide a service (for example, asking a link server to provide a menu of relevant links) expressed in an HTTP link, whereby the HTTP link described an item and provided some context within which it was referenced. Part 1 formalizes the expression of the item description, its context, and the service being requested. The definitions of all concepts are separated from their representation and the protocol by which the representations are transported.

Very few bounds are placed on how ContextObjects can be extended or applied. The Committee did not want to prescribe the limits on what kinds of creative applications there might be for ContextObjects and the OpenURL Framework in other communities. For example, the Committee could imagine storing ContextObjects in databases or using ContextObjects as the containers to transfer item and context information between servers in a web services environment. Two linking servers could talk to one another by exchanging ContextObjects. In the latter scenario, ContextObjects might be transported using an XML-based protocol such as SOAP. OpenURL 0.1 did not provide such capabilities.

Such abstraction and open-endedness is sometimes disconcerting for the development community. System providers may be reluctant to invest development resources if they fear that the Standard is

too general to be able to create interoperable solutions or that the Standard may change without their knowledge or involvement. To address these issues, the Committee introduced the notions of the OpenURL Framework Registry, OpenURL Framework Application, and Community Profile.

The OpenURL Framework Registry <<http://www.openurl.info/registry>> provides a mechanism for the public disclosure of specific selections for the representation and transportation of ContextObjects. For the purpose of this discussion, these selections will be described as registered entries. Each registered entry is assigned a unique identifier so it may be referenced unambiguously.

An OpenURL Framework Application is one instantiation of the OpenURL Framework meant for a specific community of adopters in a particular application domain. In essence, Part 1 specifies how a community can define their own OpenURL Framework Application. In general terms, a community defines an OpenURL Framework Application by selecting entries from the Registry it needs to represent and transport ContextObjects. If necessary and/or desired, the community may define new entries and, subject approval of Registry administrators, register these new entries.

A Community Profile is an unambiguous summary of one OpenURL Framework Application. For an implementer, the Community Profile unambiguously specifies the scope and boundaries of compliance by listing a selection of registered entries that OpenURL Framework Applications within that community are expected to support. To prevent a given Community Profile from becoming a moving target for a developer, the Committee envisions the Registry being under strict version control. When a community chooses to evolve its OpenURL Framework Application, it develops a new Community Profile. It may have to create new entries, register them, and have new unique identifiers assigned to them.

Part 2 (Sections 12 through 15) defines a ContextObject Format inspired by the query string of the HTTP(S) GET request as specified in OpenURL 0.1. The Key/Encoded-Value or KEV ContextObject Format defines how to represent a ContextObject as a concatenation of ampersand-delimited Key/Encoded-Value pairs. The foremost purpose of the KEV ContextObject Format is backward compatibility. It provides an elegant transition from the OpenURL 0.1 specification to this Standard.

Part 3 (Sections 16 through 19) defines a ContextObject Format based on XML (eXtensible Markup Language). XML Documents are widely used in the exchange of structured text and data between computer applications. The XML ContextObject Format is about the future. Using the full expressive power of the XML syntax, ContextObjects can convey greater detail, which Resolvers can use to provide more appropriate services.

Part 4 (Sections 20 through 22) specifies mechanisms by which ContextObject Representations can be transported using the HTTP(S) protocol. Collectively, these are called OpenURL Transports.

Parts 2, 3, and 4 define the initial content of the OpenURL Framework Registry, which is sufficient to deploy two OpenURL Framework Applications. These two Applications are defined by two Community Profiles: the Level 1 San Antonio Community Profile (SAP1) and the Level 2 San Antonio Community Profile (SAP2). They are defined, respectively, in Appendix C and Appendix D.

SAP1 formalizes the OpenURL 0.1 specification under the new Standard and adds a few of the enhancements requested by the scholarly-information community. To further assist developers in transitioning their existing application to one that complies with SAP1, a set of Implementation Guidelines have been created to provide step-by-step instructions <http://www.openurl.info/registry/docs/implementation_guidelines>.

To illustrate how SAP1 retains the original simplicity of OpenURL 0.1, let us convert the OpenURL 0.1 sample given earlier to an OpenURL that conforms with the OpenURL Framework Application defined by SAP1:

```
http://www.example.com/resolver?url_ver=Z39.88-2004
&url_ctx_fmt=info:ofi/fmt:kev:mtx:ctx
&rft_val_fmt=info:ofi/fmt:kev:mtx:journal
```

```

&rft.genre=article
&rft.atitle=p27-p16 Chimera: A Superior Antiproliferative
&rft.jtitle=Molecular Theory
&rft.aulast=McArthur
&rft.aufirst=James
&rft.date=2001
&rft.volume=3
&rft.issue=1
&rft.spage=8
&rft.epage=13

```

Three new tags have been added for version control and format declaration (to describe to the linking server what follows), and prefixes have been added to the tags to avoid ambiguity.

With tools like the Implementation Guidelines, the San Antonio Profiles, and the OpenURL Framework Registry, the OpenURL Framework Standard extends the ideas underlying the original OpenURL to new and creative uses, while it retains the simplicity of its predecessor.

Technical Considerations

Recognizing the international environments in which ContextObjects will be used, the Committee selected Unicode as the abstract character repertoire for ContextObjects. Without excluding other encoding forms, the Committee selected UTF-8 as the default encoding form of the Unicode Coded Character Set.

This Standard originated in the scholarly-information community for the purpose of providing context-sensitive linking services. The significantly more general OpenURL Framework is a reflection of NISO's charge to the Committee to develop an extensible Standard. Extensibility is implemented through the OpenURL Framework Registry. Initially, this Registry contains entries that support the creation of OpenURL Framework Applications in the scholarly-information community. However, other user communities may add new entries to support different applications. The Registry records the following:

- To support the representation of ContextObjects and the resources of which ContextObjects convey descriptions:
 - Character Encodings
 - Formats to express ContextObjects, including the Serializations, Constraint Languages, and Constraint Definitions used by those Formats. For example, the XML ContextObject Format uses XML as its Serialization and is constrained by an XML Schema.
 - Namespaces used to identify resources of which ContextObjects contain descriptions
 - Metadata Formats used to represent particular classes of resources of which ContextObjects contain descriptions
- Methods to transport ContextObject Representations
- Community Profiles that list selections of the above made by specific communities for their OpenURL Framework Application

The initial Registry contains two Formats to express ContextObjects: the Key/Encoded-Value Format and the XML Format. Communities may define and register new ContextObject Formats, thereby enabling the creation of new OpenURL Framework Applications. The initial Registry also contains a suite of HTTP(S)-based methods to transport representations of ContextObjects. Two Community Profiles are included in the initial content of the Registry. The Committee created these

to provide support for the existing OpenURL 0.1 application as used in the scholarly-information community under this Standard.

The Level 1 San Antonio Community Profile (SAP1): A Community Profile that is based on the Key/Encoded-Value Format to represent ContextObjects. It uses Namespaces and Metadata Formats that are important to the scholarly-information community. In the definition of this Community Profile, care has been taken to provide a certain level of backward compatibility with the OpenURL 0.1 specification, while at the same time providing enhanced capabilities to describe referenced resources and the network context in which the references occur.

The Level 2 San Antonio Community Profile (SAP2): A Community Profile that is based on the XML Format to represent ContextObjects. It uses Namespaces and Metadata Formats that are important to the scholarly-information community. It introduces a new level of expressiveness to describe referenced resources and the network context in which the references occur.

Reaffirmation

This standard underwent a periodic review in 2009-2010. It was approved for reaffirmation by NISO on February 12, 2012 and by ANSI on May 13, 2010.

Suggestions for improving this standard are welcome. They should be sent to the National Information Standards Organization, 3600 Clipper Mill Road, Suite 302, Baltimore, MD 21211, telephone (301) 654 2512.

Trademarks, Service Marks

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

This Standard was processed and approved for submittal to ANSI by the National Information Standards Organization. It was balloted by the NISO Voting Members January 26, 2004-March 10, 2004. This Standard will be up for review in 2009. Suggestions for improving this Standard are welcome. They should be sent to the National Information Standards Organization, 4733 Bethesda Avenue, Suite 300, Bethesda, MD 20814. NISO approval of this Standard does not imply that all Voting Members voted for its approval.

Disclaimer

Use of this Standard is voluntary. NISO accepts no responsibility for any loss or damage caused to any person or organization as result of any error, omission, or misleading statement in the information presented in this Standard or due to implementing this Standard.

NISO Voting Members

At the time it approved the Standard in 2005, NISO had the following Voting Members:

3M

Susan Boettcher, John Nelson (Alt)

American Association of Law Libraries

Robert L. Oakley, Mary Alice Baish (Alt)

American Chemical Society

tba

American Library Association

Betty Landesman

American Society for Information Science and Technology (ASIS&T)

Gail Thornburg

American Society of Indexers

Judith Gibbs

American Theological Library Association

Myron Chace

ARMA International

Diane Carlisle

Armed Forces Medical Library

Diane Zehnpfennig, Emily Court (Alt)

Art Libraries Society of North America (ARLIS/NA)

Sarah McCleskey

AIIM International

Betsy A. Fanning

Association of Information and Dissemination Centers (ASIDIC)

Marjorie Hlava

Association of Jewish Libraries

Caroline R. Miller, Elizabeth Vernon (Alt)

Association of Research Libraries (ARL)

Duane E. Webster, Julia Blixrud (Alt)

Auto-Graphics, Inc.

Paul Cope

Barnes & Noble, Inc.

Douglas Cheney

Book Industry Communication

Brian Green

California Digital Library

Daniel Greenstein, John Kunze (Alt)

Cambridge Information Group

Michael Cairns, Matthew Dunie (Alt)

College Center for Library Automation (CCLA)

J. Richard Madaus, Ann Armbrister (Alt)

Colorado State Library

Brenda Bailey-Hainer, Steve Wrede (Alt)

CrossRef

Edward Pentz, Amy Brand (Alt)

Davandy, L.L.C.

Michael J. Mellinger

Docutek Information Systems

Philip Kesten, Slaven Zivkovic (Alt)

Dynix Corporation

Lynn Thackeray, Gail Wanner (Alt)

EBSCO Information Services

Gary Coker, Oliver Pesch (Alt)

Elsevier Science Inc.

Anthony Ross, John Mancina (Alt)

Endeavor Information Systems, Inc.

Verne Coppi, Cindy Miller (Alt)

Entopia, Inc.

Igor Perisic

Ex Libris

James Steenbergen

Fretwell-Downing Informatics

Matthew Goldner, Robin Murray (Alt)

Gale Group

Katherine Gruber, Justine Carson (Alt)

Geac Library Solutions

Eric Conderaerts, Eloise Sullivan (Alt)

GIS Information Systems, Inc.

Candy Zemon, Paul Huf (Alt)

H.W. Wilson Company

Ann Case, Patricia Kuhr (Alt)

Helsinki University Library

Juha Hakala

Index Data

Sebastian Hammer, David Dorman (Alt)

Information Use Management and Policy Institute/FSU

Charles McClure, John Carlo Bertot (Alt)

Infotrieve

Jan Peterson

Innovative Interfaces, Inc.

Gerald M. Kline, Betsy Graham (Alt)

Institute for Scientific Information

Carolyn Finn

The International DOI Foundation

Norman Paskin

John Wiley & Sons, Inc.

Eric A. Swanson

Library Binding Institute

Joanne Rock

Library of Congress

Sally H. McCallum

The Library Corporation

Mark Wilson, Ted Koppel (Alt)

Los Alamos National Laboratory

Richard E. Luce

Lucent Technologies

M. E. Brennan

Medical Library Association

Nadine P. Ellero, Carla J. Funk (Alt)

MINITEX

Cecelia Boone, William DeJohn (Alt)

Modern Language Association

Daniel Bokser, B. Chen (Alt)

Motion Picture Association of America

Axel aus der Muhlen

MuseGlobal, Inc.

Kate Noerr, Clifford Hammond (Alt)

Music Library Association

Mark McKnight, David Summerfield (Alt)

National Agricultural Library

Eleanor G. Frierson, Gary K. McCone (Alt)

National Archives and Records Administration
Nancy Allard

National Federation of Abstracting and Information Services (NFAIS)
Marjorie Hlava

National Library of Medicine
Betsy L. Humphreys

National Security Agency
Kathleen Dolan

Nylink
Mary-Alice Lynch, Jane Neale (Alt)

OCLC, Inc.
Larry Olszewski

Openly Informatics, Inc.
Eric Hellman

ProQuest Information and Learning
Todd Fegan, James Brei (Alt)

Random House, Inc.
Laurie Stark

Recording Industry Association of America
Linda R. Bocchi, Michael Williams (Alt)

The Research Libraries Group
Lennie Stovel, Joan Aliprand (Alt)

Serials Solutions, Inc.
Mike McCracken

SIRSI Corporation
Greg Hathorn, Slavko Manojlovich (Alt)

Society for Technical Communication (STC)
Frederick M. O'Hara, Jr.,
Annette D. Reilly (Alt)

Society of American Archivists
Lisa Weber

Special Libraries Association (SLA)
Marcia Lei Zeng

Synapse Corporation
Trish Yancey, Dave Clarke (Alt)

TAGSYS, Inc.
John Jordon, Anne Salado (Alt)

Talis Information Ltd
Terry Willan, Katie Anstock (Alt)

Triangle Research Libraries Network
Mona C. Coutts

U.S. Department of Commerce, NIST, Office of Information Services
tba

U.S. Department of Defense, DTIC (Defense Technical Information Center)
Gopalakrishnan Nair, Jane L. Cohen (Alt)

U.S. Department of Energy, Office of Scientific & Technical Information
Ralph L. Scott, Karen J. Spence (Alt)

U.S. Government Printing Office
Judith C. Russell, T.C. Evans (Alt)

U.S. National Commission on Libraries and Information Science (NCLIS)
Robert E. Molyneux

VTLS, Inc.
Carl Grant

WebFeat
Todd Miller, Paul Duncan (Alt)

ANSI/NISO Z39.88-2004 Review Voting Pool

At the time NISO approved this Standard for reaffirmation in 2010, the following were members of the Z39.88-2004 Review Voting Pool:

American Library Association
Cindy Hepfer

American Psychological Association
Janice Fleming

American Society for Information Science & Technology (ASIS&T)
Marcia Zeng

Association of Information & Dissemination Centers (ASIDIC)
Drucilla Ekwurzel

Association of Research Libraries (ARL)
Julia Blixrud

Auto-Graphics, Inc.
Mary Jackson

Bibliographical Center for Research

Brenda Bailey-Hainer

Council on Library and Information Resources (CLIR)
Amy Friedlander

CrossRef
Chuck Koscher

Dow Jones Factiva
Marti Heyman

EBSCO Information Services
Oliver Pesch

EnvisionWare, Inc.
Robert Walsh

Ex Libris, Inc.
Nettie Lagace

Inera Inc.

Bruce Rosenblum

INFLIBNET Centre

Rajesh Chandrakar

International DOI Foundation

Norman Paskin

Ithaka/JSTOR/ARTstor

Evan Owens

Library of Congress

John Zagas

MINITEX Library Information Network

Cecelia Boone

Music Library Association

Mark McKnight

National Agricultural Library

Gary McCone

National Federation of Advanced Information Services (NFAIS)

Marjorie Hlava

National Library of Medicine (NLM)

Barbara Rapp

Polaris Library Systems

Eric Graham

Serials Solutions, Inc.

Harry Kaplanian

Standards Australia

LoAnne Lam

The Library Corporation (TLC)

DJ Miller

NISO Board of Directors

At the time NISO approved this Standard in 2005, the following individuals served on its Board of Directors:

Jan Peterson, Chair

Infotrieve

Carl Grant, Vice Chair and Chair-Elect

VTLS, Inc.

Beverly P. Lynch, Immediate Past Chair

UCLA Graduate School of Education & Information Studies

Michael J. Mellinger, Treasurer

Davandy, LLC

Patricia Stevens, Chair of SDC

OCLC, Inc.

Patricia R Harris, Executive Director /

Secretary

NISO

Directors:**Brian Green**

BIC/EDItEUR

Daniel Greenstein

California Digital Library

Jose-Marie Griffiths

University of Pittsburgh

Deborah Loeding

The H. W. Wilson Company

Richard E. Luce

Los Alamos National Laboratory

Sally McCallum

Library of Congress

Oliver Pesch

EBSCO Publishing

Committee AX Members

The following individuals served on Committee AX, *The OpenURL Framework for Context-Sensitive Services* that developed the 2005 standard:

Ann Apps

The University of Manchester, UK

Oren Beit-Arie

Ex Libris (USA), Inc

Karim Boughida

Getty Research Institute

Karen Coyle

University of California

Todd Fegan

ProQuest Information and Learning

Tony Hammond

Elsevier, Ltd. and (as of April 2004) Nature Publishing Group

Eric Hellman

Openly Informatics, Inc.

Lou Knecht

National Library of Medicine

Larry Lannom

Corporation for National Research Initiatives

Cliff Morgan (NISO SDC Liaison)

John Wiley & Sons, Ltd.

Mark Needleman

SIRSI Corporation

Eamonn Neylon

Manifest Solutions

Phil Norman

OCLC, Inc.

Oliver Pesch

EBSCO Publishing

Harry Samuels

Endeavor Information Systems, Inc.

Herbert Van de Sompel

Los Alamos National Laboratory

Eric F. Van de Velde (Chair)

California Institute of Technology

Acknowledgements

Standards Committee AX gratefully acknowledges the contributions made by Herbert Van de Sompel (Ghent University, Cornell University, Los Alamos National Laboratory); Patrick Hochstenbach (Ghent University); and Oren Beit-Arie (Ex Libris USA, Inc). Their original research, visionary thinking, and crucial contributions led to the formation of Committee AX and this Standard.

The Committee thanks the following individuals for their substantial assistance to the process of creating this Standard: Mary Alice Ball, University of Chicago Press; Priscilla Caplan, Florida Center for Library Automation; Young Jun Choi, KINS, Inc. (South Korea); Thom Hickey, OCLC; Marjorie Hlava, Access Innovations, Inc.; Gail M. Hodge, CENDI; Mike Hoover, ProQuest Information and Learning; Pat Stevens, OCLC; and Jeff Young, OCLC.

The Committee also thanks the following individuals for their efforts in providing the logistical support for one or more Committee meetings: Jacqueline J. Eudell, CNI; and Patti Franklin, CNRI.

The Committee thanks the following institutions for providing meeting space and logistical support for one or more Committee meetings: the California Institute of Technology, the Corporation for National Research Initiatives, and the Getty Research Institute.

The OpenURL Framework for Context-Sensitive Services

1 Purpose and Scope

The OpenURL Framework Standard defines an architecture for creating OpenURL Framework Applications, briefly called *Applications* in the remainder of this Standard. An *Application* is a networked service environment, in which packages of information are transported over a network. These packages have a description of a referenced resource at their core, and they are transported with the intent of obtaining context-sensitive services pertaining to the referenced resource. To enable the recipients of these packages to deliver such context-sensitive services, each package describes the referenced resource itself, the network context in which the resource is referenced, and the context in which the service request takes place. These packages are *ContextObject Representations*.

Part 1 (Sections 5 through 11) defines the *ContextObject* as an abstract information construct. This Standard is independent of the application domain. It does not constrain the type of resources that may be described in a *ContextObject*. However, it does specify how communities can create concrete *ContextObject Representations* for use in their *Applications*. To that end, this Standard introduces the following core components of the OpenURL Framework: *Character Encodings*, *Serializations*, *Constraint Languages*, *ContextObject Formats*, *Metadata Formats*, and *Namespaces*.

Although *ContextObject Representations* *may* reside as autonomous data files in information systems, this Standard expects that *ContextObject Representations* will be transported between networked systems. Section 10 defines *Transports*, a core component of the OpenURL Framework in which communities specify how to transport *ContextObject Representations* in their *Applications*. This Standard does not restrict the purpose of such transportation. It is expected, however, that most transportations of *ContextObject Representations* will be requests for context-sensitive services pertaining to the referenced resource.

The targets of the transportation of *ContextObject Representations* are networked systems that are able to process *ContextObject Representations* and provide context-sensitive services. These systems are called *Resolvers*. *Resolver* behavior and usage are outside of the scope of this Standard. However, a community *may* use a *Community Profile* to define conformance for *Resolvers* that operate in its application domain. A community specifies its selections for each of the aforementioned core components in a *Community Profile*. This is the final core component of the OpenURL Framework, and it is defined in Section 11.

Section 6 defines the *OpenURL Framework Registry* and the rules that govern its usage. The *OpenURL Framework Registry* contains the selections for all core components made by communities that define *Applications*. The *Registry* ensures that this Standard can be used in many different application domains.

Parts 2, 3, and 4 specify the initial content of the *OpenURL Framework Registry* and provide detailed definitions of the registered content. The initial *Registry* deploys two *Applications* for the scholarly-information community. These *Applications* are defined by two *Community Profiles*:

- The Level 1 San Antonio Community Profile (SAP1), defined in Appendix C, is based on a Key/Encoded-Value Representation of *ContextObjects*. Key/Encoded-Value *ContextObject Representations* may be transported by any one of the three HTTP-based *Transports* defined in Part 4. The *Transport* defined in Section 22 was developed to provide a certain level of backward compatibility with the OpenURL 0.1 specification.

- The Level 2 San Antonio Community Profile (SAP2), defined in Appendix D, is based on an XML *Representation of ContextObjects*. XML *ContextObject Representations* may be transported by any one of two HTTP-based Transports defined in Part 4, Sections 20 and 21. The SAP2 *Community Profile* is not backward compatible with the OpenURL 0.1 specification.

Part 2 (Sections 12 through 15) defines a *ContextObject Format* inspired by the query string of the HTTP(S) GET request as specified in OpenURL 0.1. The *Key/Encoded-Value ContextObject Format* defines how to represent a *ContextObject* as a concatenation of ampersand-delimited Key/Encoded-Value pairs. The foremost purpose of the *Key/Encoded-Value ContextObject Format* is backward compatibility. It provides an elegant transition from the OpenURL 0.1 specification to this Standard.

Part 3 (Sections 16 through 19) defines a *ContextObject Format* based on XML (eXtensible Markup Language). XML *Documents* are widely used in the exchange of structured text and data between computer applications. The XML *ContextObject Format* is about the future. Using the full expressive power of the XML syntax, *ContextObjects* can be described in greater detail, which *Resolvers* can use to provide more and better services.

Part 4 (Sections 20 through 22) specifies mechanisms by which *ContextObject Representations* can be transported using the HTTP(S) protocol. Collectively, these are called *OpenURL Transports*.

Communities interested in deploying new *Applications* **should** use Parts 2, 3 and 4 as a guideline. Deploying a new *Application* consists of the following steps:

- Register any new definitions of the following core components of the OpenURL Framework that are needed to support the *Application*: *Character Encodings*, *Serializations*, *Constraint Languages*, *ContextObject Formats*, *Metadata Formats*, *Namespaces*, and *Transports*.
- Construct a new *Community Profile* that defines the *Application* by selecting appropriate *Registry* entries.
- Register the *Community Profile*.

Communities **should** create Implementation Guidelines to simplify implementation and deployment of their *Applications*.

2 Referenced Standards

This Standard references the following existing standards:

- [1] *Extensible Markup Language (XML) 1.0*. [online] Third edition. 4 February 2004 [online] [cited 4 November 2004] Available from World Wide Web: <<http://www.w3.org/TR/REC-xml>>
- [2] *Extensible Markup Language (XML)—XML Path Language (XPath)*. Version 1.0. 16 November 1999. [online] [cited 4 November 2004] Available from World Wide Web: <<http://www.w3.org/TR/xpath>>
- [3] *Extensible Markup Language (XML)—XML Schema Part 1: Structures*. Second edition. 28 October 2004. [online] [cited 4 November 2004] Available from World Wide Web: <<http://www.w3.org/TR/xmlschema-1/>>
- [4] *Extensible Markup Language (XML)—XML Schema Part 2: Datatypes*. Second edition. 28 October 2004. [online] [cited 4 November 2004] Available from World Wide Web: <<http://www.w3c.org/TR/xmlschema-2/>>

- [5] IETF RFC 2119, *Keywords for use in RFCs to Indicate Requirement Levels*. March 1997. [online] [cited 4 November 2004] Available from World Wide Web: <<http://www.ietf.org/rfc/rfc2119.txt>>
- [6] IETF RFC 2396, *Uniform Resource Identifiers (URI): Generic Syntax*. August 1988. [online] [cited 4 November 2004] Available from World Wide Web: <<http://www.ietf.org/rfc/rfc2396.txt>> (Draft revision of IETF RFC 2396 available from World Wide Web: <<http://www.ietf.org/internet-drafts/draft-fielding-uri-rfc2396bis-07.txt>>. New RFC number to be assigned.)
- [7] Internet Assigned Naming Authority (IANA), *List of Registered Character Sets*. Last updated 2004-02-06. [online] [cited 4 November 2004] Available from World Wide Web: <<http://www.iana.org/assignments/character-sets>>
- [8] Internet Assigned Naming Authority (IANA), *Uniform Resource Identifier (URI) Schemes*. Last updated 10 October 2004. [online] [cited 4 November 2004] Available from World Wide Web: <<http://www.iana.org/assignments/uri-schemes>>
- [9] Internet Assigned Naming Authority (IANA), *Uniform Resource Names (URN) Namespaces*. Last updated 10 October 2004. [online] [cited 4 November 2004] Available from World Wide Web: <<http://www.iana.org/assignments/urn-namespaces>>
- [10] *Data elements and interchange formats—Information interchange—Representation of dates and times*. ISO 8601:2000. Geneva: International Organization for Standardization, Switzerland, 2000.
- [11] *The Unicode Standard Version 4.0*. The Unicode Consortium. Reading, MA: Addison-Wesley, 2000. Updates available from World Wide Web: <<http://www.unicode.org/>>
- [12] *W3C Date and Time Formats*. Submitted to W3C 15 September 1997. [online] [cited 4 November 2004] Available from World Wide Web: <<http://www.w3.org/TR/NOTE-datetime>>
- [13] *Character Encoding Model*. Unicode Technical Report #17, Revision 5. 2004-09-09. [online] [cited 4 November 2004] Available from World Wide Web: <<http://www.unicode.org/unicode/reports/tr17>>
- [14] IETF RFC 2616, *Hypertext Transfer Protocol — HTTP/1.1*. June 1999. [online] [cited 4 November 2004] Available from World Wide Web: <<http://www.ietf.org/rfc/rfc2616.txt>>
- [15] *The mailto URL scheme*. IETF RFC 2368. The Internet Society, 1998. [online] [cited 4 November 2004] Available from World Wide Web: <<http://www.ietf.org/rfc/rfc2368.txt>>
- [16] *Using The ISSN (International Serial Standard Number) as URN (Uniform Resource Names) within an ISSN-URN Namespace*. IETF RFC 3044. The Internet Society, 2001. [online] [cited 4 November 2004] Available from World Wide Web: <<http://www.ietf.org/rfc/rfc3044.txt>>
- [17] Carl Lagoze, Herbert Van de Sompel, Michael Nelson, Simeon Warner. *The Open Archives Initiative Protocol for Metadata Harvesting*. Protocol version 2.0. Document version 2004/10/12T15:31:00Z. [online] [cited 4 November 2004] Available from World Wide Web: <<http://www.openarchives.org/OAI/2.0/openarchivesprotocol.htm>>
- [18] *The Dublin Core Metadata Element Set*. ANSI/NISO Z39.85-2001. Bethesda, MD: National Information Standards Organization, 2001. [online] [cited 4 November 2004] Available from World Wide Web: <<http://www.niso.org/standards/resources/Z39-85.pdf>>

3 Notational Conventions

The Internet Engineering Task Force (IETF) RFC 2119 [5] specifies the meaning of the following key words and key phrases: ***must***, ***must not***, ***required***, ***shall***, ***shall not***, ***should***, ***should not***, ***recommended***, ***may***, and ***optional***. When these appear in this Standard in a ***bold italic font style***, they have the meaning as specified by IETF RFC 2119:

- ***must***: This word, or the terms ***required*** or ***shall***, mean that the definition is an absolute requirement of the specification.
- ***must not***: This phrase, or the phrase ***shall not***, mean that the definition is an absolute prohibition of the specification.
- ***should***: This word, or the adjective ***recommended***, mean that there may exist valid reasons in particular circumstances to ignore a particular item, but the full implications must be understood and carefully weighed before choosing a different course.
- ***should not***: This phrase, or the phrase ***not recommended*** mean that there may exist valid reasons in particular circumstances when the particular behavior is acceptable or even useful, but the full implications should be understood and the case carefully weighed before implementing any behavior described with this label.
- ***may***: This word, or the adjective ***optional***, means that an item is truly optional. One vendor may choose to include the item because a particular marketplace requires it or because the vendor feels that it enhances the product while another vendor may omit the same item. An implementation which does not include a particular option ***must*** be prepared to interoperate with another implementation which does include the option, though perhaps with reduced functionality. In the same vein an implementation which does include a particular option ***must*** be prepared to interoperate with another implementation which does not include the option (except, of course, for the feature the option provides.)

Terms defined in this Standard will be shown in *italics* whenever this Standard uses them in the defined meaning, as in *ContextObject* or *By-Value Metadata*.

References to *Registry Identifiers* or portions of *Identifiers* are shown in **bold**, as in **info:doi**.

Throughout this Standard, examples are provided to support a better understanding of the key terms as they are defined. The examples are excerpts from valid *Representations* of *ContextObjects*. Many examples use an informal property-list syntax in which each property is listed on a separate line and a property consists of a key term and associated value, as in:

<key> = <value>

This property-list syntax is for illustrative purposes only. It is not part of this Standard, and it ***must not*** be used to represent *ContextObjects* in *Applications*. Parts 2 and 3 formally define two *ContextObject Formats* (KEV and XML). Only *ContextObject Formats* that are formally defined in the *OpenURL Framework Registry* are available for use in an *Application*.

Tables that specify constraints use the following short-hand notation:

Constraint in short-hand	Minimum Occurrence	Maximum Occurrence
0	0	0
1	1	1
≥0	0	unbounded
≤1	0	1
≥1	1	unbounded

Some Section titles end with the suffix “[Registry]”, as in “Section 7.1 Serializations [Registry]”. These Sections define core components of the OpenURL Framework, and instantiations of these core components **must** be registered.

This Standard uses many *Identifiers* based on the “info” URI scheme. On June 19th, 2003, representatives from NISO, NISO Committee AX, the IETF, and the W3C met to discuss the identification of resources in the OpenURL Framework. There was a consensus to proceed with the registration of a new top-level URI scheme. The first Internet-Draft for the “info” URI scheme was published on September 25th, 2003. A revision was published on December 5th, 2003. These drafts and any subsequent versions will be maintained on the <http://info-uri.info/> website. At the time of writing this Standard, the “info” URI scheme is awaiting approval by the Internet Engineering Steering Group (IESG) for publication as an informational RFC.

4 Definitions

The following terms when italicized in this Standard have the meanings indicated here:

Term	Definition
(OpenURL Framework) <i>Application</i>	A networked service environment for the transportation of <i>ContextObject Representations</i> . The core characteristics of an <i>Application</i> are specified in a <i>Community Profile</i> .
<i>By-Reference Metadata</i>	A <i>Descriptor</i> that details properties of an <i>Entity</i> by the combination of: (1) a URI reference to a <i>Metadata Format</i> and (2) the network location of a particular instance of metadata about the <i>Entity</i> , the metadata being expressed according to the indicated <i>Metadata Format</i> .
<i>By-Reference OpenURL Transport</i>	A <i>Transport</i> that uses either the HTTP or the HTTPS network protocol for conveying over a network the reference to a <i>ContextObject Representation</i> . This reference is contained in the value associated with a single key within a query string, which is transported using either a GET or POST method.
<i>By-Value Metadata</i>	A <i>Descriptor</i> that specifies properties of an <i>Entity</i> by the combination of: (1) a URI reference to a <i>Metadata Format</i> ; and (2) a particular instance of metadata about the <i>Entity</i> , expressed according to the indicated <i>Metadata Format</i> .
<i>By-Value OpenURL Transport</i>	A <i>Transport</i> that uses either the HTTP or the HTTPS network protocol for conveying over a network <i>ContextObject Representations</i> . The <i>Representation</i> is contained in the value associated with a single key within a query string, which is transported using either a GET or POST method.

Term	Definition
<i>Character Encoding</i>	The combination of a character repertoire and an encoding form; a core component of the OpenURL Framework.
<i>Community Profile</i>	The definition of an <i>Application</i> as a list of selections for all core components of the OpenURL Framework; a core component of the OpenURL Framework.
<i>Context</i>	The network environment in which a <i>Referent</i> is referenced and in which a service request pertaining to the <i>Referent</i> takes place. In the <i>ContextObject</i> , the <i>Context</i> is expressed by five <i>Entities</i> : the <i>ReferringEntity</i> , the <i>Requester</i> , the <i>ServiceType</i> , the <i>Resolver</i> , and the <i>Referrer</i> .
<i>ContextObject</i>	An information construct that binds a description of a primary <i>Entity</i> — the referenced resource — together with descriptions of <i>Entities</i> that indicate the <i>Context</i> .
<i>ContextObject Format</i>	A <i>Format</i> to represent <i>ContextObjects</i> ; a core component of the OpenURL Framework.
<i>ContextObject Representation</i>	The <i>Representation</i> of a <i>ContextObject</i> according to a <i>ContextObject Format</i> .
<i>Constraint Definition</i>	A <i>Constraint Definition</i> specifies syntactic and semantic constraints for the representation of a given class of resources. The constraints are specified using a <i>Constraint Language</i> .
<i>Constraint Language</i>	A formalism used to specify syntactic and semantic restrictions on information constructs of a given class; a core component of the OpenURL Framework.
<i>Descriptor</i>	A <i>Descriptor</i> specifies information about an <i>Entity</i> using one of the following four methods: <i>Identifier</i> , <i>By-Reference Metadata</i> , <i>By-Value Metadata</i> , or <i>Private Data</i> .
<i>Entity</i>	One of the six possible constituents of a <i>ContextObject</i> : <i>Referent</i> , <i>Requester</i> , <i>Referrer</i> , <i>Resolver</i> , <i>ReferringEntity</i> , or <i>ServiceType</i> .
<i>Format</i>	A concrete method of expression for a class of information constructs. It is a triple comprising: (1) a <i>Serialization</i> , (2) a <i>Constraint Language</i> , and (3) a <i>Constraint Definition</i> expressed in that <i>Constraint Language</i> .
<i>Identifier</i>	A <i>Descriptor</i> that unambiguously specifies an <i>Entity</i> by means of a URI.

Term	Definition
<i>Inline OpenURL Transport</i>	A <i>Transport</i> that uses either the HTTP or the HTTPS network protocol for conveying over a network the <i>Representation</i> of one, and only one, <i>ContextObject</i> . This <i>Representation</i> consists of multiple key/value pairs within a query string, which is transported using either a GET or POST method.
<i>KEV ContextObject Format</i>	A <i>ContextObject Format</i> to represent one, and only one, <i>ContextObject</i> as a string of ampersand-delimited pairs, each pair consisting of a key and an associated value that is URL encoded.
<i>KEV ContextObject (Representation)</i>	A <i>Representation</i> of a <i>ContextObject</i> that conforms to the <i>KEV ContextObject Format</i> .
<i>KEV Metadata Format</i>	A <i>Metadata Format</i> to represent an <i>Entity</i> as a string of ampersand-delimited pairs, each pair consisting of a key and an associated value that is URL encoded.
<i>KEV Metadata (Representation)</i>	A <i>Representation</i> of an <i>Entity</i> that conforms to a <i>KEV Metadata Format</i> .
<i>KEV Serialization</i>	A method to hold in storage, or transmit over a network, the values within an information construct as a string of ampersand-delimited pairs, each pair consisting of a key and an associated value that is URL encoded.
<i>Metadata Format</i>	A <i>Format</i> to create a <i>By-Reference Metadata Descriptor</i> or a <i>By-Value Metadata Descriptor</i> of an <i>Entity</i> ; a core component of the OpenURL Framework.
<i>Namespace</i>	The set of all Uniform Resource Identifiers that comply with a specific URI scheme or a specific URN namespace; a core component of the OpenURL Framework.
<i>Private Data</i>	A <i>Descriptor</i> that specifies information about an <i>Entity</i> using a method not defined in this Standard.
<i>Referent</i>	A resource that is referenced on a network, and about which the <i>ContextObject</i> is created; an <i>Entity</i> of the <i>ContextObject</i> .
<i>Referrer</i>	The resource that generates the <i>ContextObject</i> ; an <i>Entity</i> of the <i>ContextObject</i> .
<i>ReferringEntity</i>	The resource that references the <i>Referent</i> ; an <i>Entity</i> of the <i>ContextObject</i> .

Term	Definition
<i>(OpenURL Framework) Registry</i>	The <i>Registry</i> provides a mechanism to record and publicize details of the core components of the OpenURL Framework: <i>Namespaces</i> , <i>Character Encodings</i> , <i>Serializations</i> , <i>Constraint Languages</i> , <i>ContextObject Formats</i> , <i>Metadata Formats</i> , <i>Transports</i> , and <i>Community Profiles</i> .
<i>Registry Identifier</i>	A unique name assigned on registration to specific <i>Namespaces</i> , <i>Character Encodings</i> , <i>Serializations</i> , <i>Constraint Languages</i> , <i>ContextObject Formats</i> , <i>Metadata Formats</i> , <i>Transports</i> , and <i>Community Profiles</i> .
<i>Representation</i>	A sequence of bytes that represents a resource according to a <i>Format</i> .
<i>Requester</i>	The resource that requests services pertaining to the <i>Referent</i> ; an <i>Entity</i> of the <i>ContextObject</i> .
<i>Resolver</i>	The resource at which a service request pertaining to the <i>Referent</i> is targeted; an <i>Entity</i> of the <i>ContextObject</i> .
<i>Serialization</i>	A method to hold in storage or transmit over a network the values within an information construct; a core component of the OpenURL Framework.
<i>ServiceType</i>	The resource that defines the type of service (pertaining to the <i>Referent</i>) that is requested; an <i>Entity</i> of the <i>ContextObject</i> .
<i>Transport</i>	A network protocol and the method in which it is used to convey <i>ContextObject Representations</i> ; a core component of the OpenURL Framework.
<i>XML ContextObject Format</i>	A <i>ContextObject Format</i> to represent one or more <i>ContextObjects</i> as an <i>XML Document</i> .
<i>XML ContextObject (Representation)</i>	A <i>ContextObject Representation</i> that conforms to the <i>XML ContextObject Format</i> .
<i>XML Document</i>	A sequence of bytes that satisfies the validity requirements of the Extensible Markup Language (XML) 1.0 (Second Edition) W3C Recommendation [1].
<i>XML Metadata (Representation)</i>	A <i>Representation</i> of an <i>Entity</i> that conforms to an <i>XML Metadata Format</i> .
<i>XML Serialization</i>	The method of using an <i>XML Document</i> and <i>XML Format</i> to represent a <i>ContextObject</i> .

The OpenURL Framework for Context-Sensitive Services

Part 1: ContextObjects and Transports

Part 1 (Sections 5 through 11) defines the core components of the OpenURL Framework: *Namespaces*, *Character Encodings*, *Serializations*, *Constraint Languages*, *ContextObject Formats*, *Metadata Formats*, *Transports*, and *Community Profiles*. Instances of these core components are preserved and made publicly available in the *OpenURL Framework Registry*. A community that wishes to create a new *OpenURL Framework Application* **must** create a new *Community Profile*. In this *Community Profile*, the community **must** specify instances for all core components, except the new *Community Profile* itself. If the *Registry* does not contain an instance of a core component needed by an *Application*, it is necessary to define and register an appropriate instance of the core component. The registration of a component makes this instance available for this or any other future *Application*. For example, if an *Application* needs to use a particular *Namespace*, that *Namespace* **must** be registered. Once registered, any *Application* **may** select this *Namespace* in its *Community Profile*.

Section 5 defines the *ContextObject* as an abstract information construct that consists of six *Entities*: *Referent*, *ReferringEntity*, *Requester*, *ServiceType*, *Resolver*, and *Referrer*. Each of these *Entities* is described using one or more *Descriptors*. There are four *Descriptor* types: *Identifier*, *By-Value Metadata*, *By-Reference Metadata*, and *Private Data*.

Section 6 defines the *OpenURL Framework Registry* and the rules that govern its usage. The *Registry* contains all instances of core components created by communities that deployed *Applications*. The *Registry* ensures that this Standard can be used in many different application domains.

Sections 7, 8, and 9 provide the framework for developing, defining, and registering methods to represent *ContextObjects* as character strings. Section 7 defines a *Format* as a triple consisting of a *Serialization*, a *Constraint Language*, and a *Constraint Definition*. Section 8 introduces *ContextObject Formats* and *Character Encodings* to represent *ContextObjects*. Section 9 introduces *Metadata Formats* and *Namespaces* to represent *Entities*.

Although *ContextObject Representations* **may** reside as autonomous data files in information systems, this Standard expects that *ContextObject Representations* will be transported between networked systems. Section 10 defines *Transports*, a core component of the OpenURL Framework. In a *Transport*, a community specifies how to convey over a network *ContextObject Representations*. This Standard does not restrict the purpose of the *Transport*. It is expected, however, that most *Transports* will be requests for context-sensitive services pertaining to the referenced resource (the *Referent*) and will be targeted at *Resolvers*.

A community specifies its selections for each of the core components in a *Community Profile*. This final core component of the OpenURL Framework is defined in Section 11.



5 ContextObject, Entity, and Descriptor

This Section defines the fundamental data structure of the OpenURL Framework Standard: the *ContextObject*. While this Standard does not restrict the use of the *ContextObject* to any particular environment or application, it was constructed to enable the delivery of context-sensitive services in a networked environment such as the Web.

The following scenario is used throughout this Section:

Caltech has an institutional linker server with URI <http://links.caltech.edu/menu>.

Jane Doe, a Caltech student with e-mail address jane.doe@caltech.edu, reads the following electronic scholarly article in the Elsevier ScienceDirect® collection:

McArthur, James G. et al. 2001. p27-p16 Chimera: A Superior Antiproliferative for the Prevention of Neointimal Hyperplasia. *Molecular Therapy*. 3(1) 8-13.
<doi:10.1006/mthe.2000.0239>

In the reference list of that article, she comes across a reference to the following article:

Bergelson, J. 1997. Isolation of a common receptor for coxsackie B viruses and adenoviruses 2 and 5. *Science*. (275) 1320-1323. <doi:10.1126/science.275.5304.1320> <pmid:9036860>

In this example, Jane Doe wants services for the Bergelson article, to which she found a reference in ScienceDirect®. Jane Doe might want the full text of the article. The full text may be available from ScienceDirect® itself, an aggregator, or Caltech's interlibrary-loan department. The full-text service therefore depends on the identity and affiliation of the Jane Doe, which are part of the context of the reference to the Bergelson article. In other cases, different contextual information may be important.

The *ContextObject* data structure captures relevant information for the delivery of context-sensitive services pertaining to a referenced resource. Based on a study of real-world OpenURL 0.1 usage, the Committee included the following in the *ContextObject* data structure:

- a description of the referenced resource itself (the Bergelson article),
- a description of the resource that makes the reference (the McArthur article), and
- a description of four other resources that are useful in fulfilling service requests pertaining to the referenced resource:
 1. the agent requesting the service (Jane Doe),
 2. the type of service that is requested (full text),
 3. the system at which the service request is targeted (Caltech linking server), and
 4. the system where the service request originates (ScienceDirect®).

The formal definition of *ContextObject* follows.

5.1 ContextObject and Entity

A *ContextObject* is a data structure that binds together descriptions of:

- A *Referent*: A resource that is referenced on a network and about which the *ContextObject* is created
- A *ReferringEntity*: The resource that references the *Referent*
- A *Requester*: The resource that requests services pertaining to the *Referent*

- A *ServiceType*: The resource that defines the type of service (pertaining to the *Referent*) that is requested
- A *Resolver*: The resource at which a service request pertaining to the *Referent* is targeted
- A *Referrer*: The resource that generates the *ContextObject*

The *ContextObject* is created to enable the delivery of services pertaining to the *Referent*, which is at the core of the *ContextObject*. The descriptions of the *ReferringEntity*, the *Requester*, the *ServiceType*, the *Resolver*, and the *Referrer* express the *Context* in which the *Referent* is referenced and in which the request for services pertaining to the *Referent* takes place.

The remainder of this Standard uses the term *Entity* to refer to any of the six types of resources that **may** be described in a *ContextObject*.

Example 1 uses the scenario introduced above to illustrate all *Entities* of the *ContextObject*.

Example 1: Examples of Entities

Entity	Example
<i>Referent</i>	The scholarly article by Bergelson
<i>ReferringEntity</i>	The scholarly article by McArthur
<i>Requester</i>	Jane Doe
<i>ServiceType</i>	Full text of the Bergelson article
<i>Resolver</i>	The Caltech linking server
<i>Referrer</i>	Elsevier's ScienceDirect®

5.2 Descriptor

A *Descriptor* specifies information about an *Entity*. This Section defines the four types of *Descriptors* that are available in this Standard: *Identifier*, *By-Value Metadata*, *By-Reference Metadata*, and *Private Data*.

5.2.1 Identifier

An *Identifier Descriptor* unambiguously specifies the *Entity* by means of a Uniform Resource Identifier (URI). This URI either points to the *Entity* itself or to metadata that specify the *Entity*.

Example 2: Identifiers for a Referent, Requester, and Resolver

```
rft_id = info:doi/10.1126/science.275.5304.1320
rft_id = info:pmid/9036860
req_id = mailto:jane.doe@caltech.edu
res_id = http://links.caltech.edu/menu
```

Example 2 shows *Identifier Descriptors* for a *Referent* (the Bergelson article), a *Requester* (Jane Doe), and a *Resolver* (the Caltech linking server) using the informal property-list syntax of Section 3. The key names (**rft_id**, for example) resemble those introduced in Part 2 of this Standard. However, Part 1 of this Standard uses these names for illustration only and does not formally define them.

The Digital Object Identifier (DOI) **10.1126/science.275.5304.1320** identifies the Bergelson article. As such, the URI **info:doi/10.1126/science.275.5304.1320** is an *Identifier Descriptor* for the *Referent*. The PubMed identifier **9036860** identifies metadata for the Bergelson article. Therefore, the URI **info:pmid/9036860** is also an *Identifier Descriptor* for the *Referent*.

The e-mail address **jane.doe@caltech.edu** describes Jane Doe. The corresponding URI **mailto:jane.doe@caltech.edu** is an *Identifier Descriptor* for this *Requester*.

The URI **http://links.caltech.edu/menu** describes the institutional linking server at Caltech and is an *Identifier Descriptor* for this *Resolver*.

5.2.2 By-Value Metadata

A *By-Value Metadata Descriptor* specifies properties of the *Entity* by the combination of: (1) a URI reference to a *Metadata Format*; and (2) a particular instance of metadata about the *Entity* expressed according to this *Metadata Format*.

Example 3: By-Value Metadata for a Referent

```
rft_val_fmt = info:ofi/fmt:kev:mtx:journal
rft.aulast = Bergelson
rft.auinit = J
rft.date = 1997
rft.atitle = Isolation of a common receptor for coxsackie B viruses and
adenoviruses 2 and 5
rft.jtitle = Science
rft.volume = 275
rft.spage = 1320
rft.epage = 1323
```

Example 3 shows a *By-Value Metadata Descriptor* for a *Referent*, the Bergelson article.

The URI specified as the value of the **rft_val_fmt** key (**info:ofi/fmt:kev:mtx:journal**) identifies the *Metadata Format* used to describe the Bergelson article. Sections 6, 7, and 8 explain how to construct and interpret *Registry Identifiers* for *Formats*, such as **info:ofi/fmt:kev:mtx:journal**. This particular *Registry Identifier* identifies a *Metadata Format* for a journal article.

The remaining lines in Example 3 are metadata properties for the *Referent*. The metadata keys (**aulast**, **auinit**, **date**, etc.) are from the identified *Metadata Format* for a journal article. The metadata keys are prefixed with **rft.** to indicate that the metadata describe the *Referent*.

5.2.3 By-Reference Metadata

A *By-Reference Metadata Descriptor* specifies properties of the *Entity* by the combination of: (1) a URI reference to a *Metadata Format*; and (2) the network location — specified by means of a URI — of a particular instance of metadata about the *Entity* expressed according to this *Metadata Format*.

Example 4: By-Reference Metadata for a Requester

```
req_ref_fmt = http://lib.caltech.edu/mxt/ldap.html
req_ref = ldap://ldap.caltech.edu:389/janed
```

Example 4 shows a *By-Reference Metadata Descriptor* for a *Requester*, Jane Doe. The value associated with the **req_ref** key is a pointer to (or network location of) Jane Doe's entry in the Caltech LDAP directory server. The value of the **req_ref_fmt** key specifies the *Metadata Format* of the document to which the value of the **req_ref** key points.

5.2.4 Private Data

A *Private Data Descriptor* specifies information about the *Entity* using a method not defined in this Standard. This Standard does not provide any global mechanisms to interpret *Private Data*. Instead, it is assumed that the *Resolver* and the *Referrer* have a common understanding, based on a tacit or

explicit bilateral agreement. To make it possible for the *Resolver* to interpret *Private Data*, a *ContextObject* that contains *Private Data* **should** identify the *Referrer* that created it.

Example 5: Private Data for a Referent

```
rft_dat = cites/8///citedby/12
rfr_id = info:sid/elsevier.com:ScienceDirect
```

Example 5 shows a *Private Data Descriptor* for a *Referent*. The value associated with the **rft_dat** key, **cites/8///citedby/12**, is *Private Data* provided about the *Referent*. The value associated with the **rfr_id** key, **info:sid/elsevier.com:ScienceDirect**, is an *Identifier Descriptor* of the *Referrer*. Knowing the identity of the *Referrer* might help the *Resolver* to interpret the *Private Data*.

5.3 Constraints

The number of occurrences of each *Entity* that **may** be present in a *ContextObject* is constrained:

- A *ContextObject* **must** contain exactly one *Referent*.
- A *ContextObject* **may** contain at most one *ReferringEntity*, *Requester*, and *Referrer*.
- A *ContextObject* **may** contain zero or more *ServiceTypes* and *Resolvers*.

These fundamental constraints are summarized in the first two columns of Table 1.

The remaining columns of Table 1 indicate that:

- All four *Descriptors* **may** be used to describe each of the *Entities*.
- Each type of *Descriptor* **may** be used zero or more times for the description of a specific *Entity*. No ordering or priority is defined for multiple *Descriptors*.

Table 1: Fundamental ContextObject Constraints

Entity	Number	Descriptors			
		Identifier	By-Value Metadata	By-Reference Metadata	Private Data
<i>Referent</i>	1	≥ 0	≥ 0	≥ 0	≥ 0
<i>ReferringEntity</i>	≤ 1	≥ 0	≥ 0	≥ 0	≥ 0
<i>Requester</i>	≤ 1	≥ 0	≥ 0	≥ 0	≥ 0
<i>ServiceType</i>	≥ 0	≥ 0	≥ 0	≥ 0	≥ 0
<i>Resolver</i>	≥ 0	≥ 0	≥ 0	≥ 0	≥ 0
<i>Referrer</i>	≤ 1	≥ 0	≥ 0	≥ 0	≥ 0

ContextObject *Formats* that define specific methods to represent *ContextObjects* (see Section 8.2) **must not** relax the constraints expressed in Table 1, but they **may** restrict them:

- A *ContextObject* *Format* **may** limit the number of occurrences of *ServiceType* and *Resolver* *Entities*.
- A *ContextObject* *Format* **may** limit the number of *Descriptors* that **may** be used to describe each *Entity*.
- A *ContextObject* *Format* **must not** allow multiple *Referent*, *ReferringEntity*, *Requester*, or *Referrer* *Entities*.

Any additional constraints **must** be specified in *ContextObject Format* definitions.

When one *Entity* of a *ContextObject* is described by multiple *Descriptors*, those *Descriptors* **must** describe the same resource. For example, when a *ContextObject* contains two *Identifier Descriptors* and one *By-Value Metadata Descriptor* for one *Referent Entity*, all three *Descriptors* **must** describe the same *Referent*.

When there are multiple occurrences of the same *Entity* in one *ContextObject*, each occurrence **must** represent a different resource. These multiple occurrences **must not** be interpreted as variant descriptions of the same resource. For example, two *Resolver Entities* contained in one *ContextObject* **must** represent two distinct *Resolvers*. In this case, each *Resolver Entity* **may** have multiple *Descriptors*, each of which **must** be a variant description of the same *Resolver*.

ContextObject Formats that allow multiple occurrences of *ServiceType* and/or *Resolver Entities* **must** define how multiple *Descriptors* are grouped to bind to particular *Entities*.

The basic data model for *ContextObjects* does not constrain the number of *ContextObjects* that **may** be represented in an instance document that conforms to the *ContextObject Format*. *ContextObject Formats* **may** constrain this number, and each *Community Profile* provides this information (see Sections 11, 15, and 19).

6 Registry

This Section defines the OpenURL Framework Registry, referred to as the *Registry* in the remainder of this Standard. It is based at <<http://www.openurl.info/registry>>.

Upon approval of this Standard, NISO will establish one or more Maintenance Agencies for the *Registry*. The responsibilities and duties for Maintenance Agencies of the OpenURL Framework Standard are specified in Appendix A.

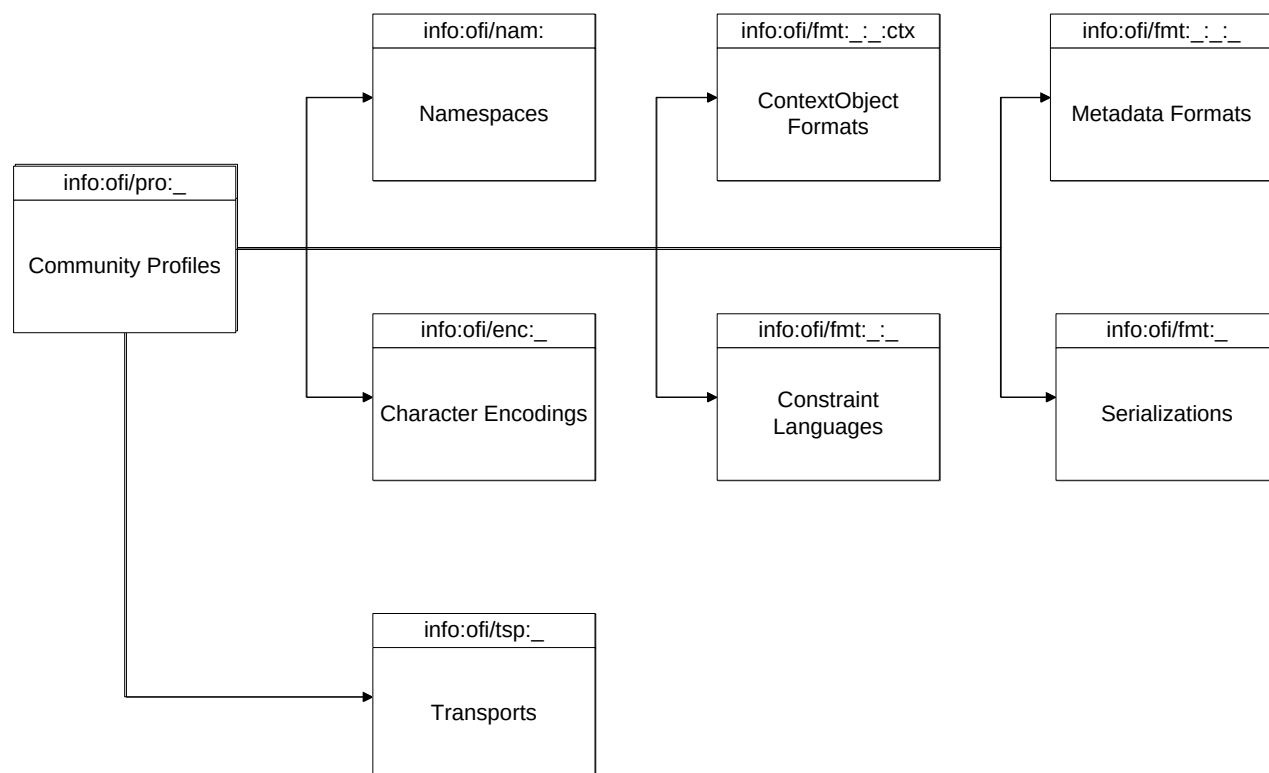
6.1 Registry Entries

When a community defines an *Application*, it **must** specify selections for each of the core components of the OpenURL Framework:

- For representing *ContextObjects*:
 - *Character Encodings* (one or more)
 - *Serializations* (one)
 - *Constraint Languages* (one)
 - *ContextObject Formats* (one)
- For representing *Entities* of *ContextObjects*:
 - *Namespaces* (zero or more)
 - *Metadata Formats* (zero or more)
- For transporting *ContextObject Representations*:
 - *Transports* (one or more)
- For defining *Applications*:
 - *Community Profiles* (one)

Figure 1 illustrates the structure of the *Registry* and shows how *Community Profiles* define the characteristics of an *Application* by listing community-specific selections for the core components of the OpenURL Framework.

Figure 1: Core Components of the OpenURL Framework



Example 6: A Registry Entry

info:ofi/nam:info:doi:	
dc:title	Namespace for Digital Object Identifiers
dc:creator	International DOI Foundation
dc:date	2004-01-01
dc:identifier	http://errol.oclc.org/info-uri.info/info:doi/?metadataPrefix=reg
dc:identifier	http://errol.oclc.org/info-uri.info/info:doi/.reg
dc:identifier	info:doi/

Example 6 shows a *Registry* entry that describes an instance of a *Namespace*, a core component of the OpenURL Framework introduced in Section 9.1. This entry describes the *Namespace* of Digital Object Identifiers (DOI), which is introduced in Appendix C. The top row displays the *Registry Identifier* of the *Registry* entry, and remaining rows use Dublin Core metadata [18] to describe the *Registry* entry (see Sections 6.2 and 6.3).

Part 1 is concerned with the OpenURL Framework; it does not register specific instances of core components. In Parts 2, 3, and 4 and Appendices Appendix B, Appendix C, and Appendix D, this Standard defines, registers, and uses specific instances of core components. However, *Registry* entries in this Standard are provided for illustrative purposes only and are often only partially displayed. For example, the **dc:date** field is usually omitted as it is 2004-01-01 for all entries in the

initial *Registry*. The authoritative *Registry* entries are in the online *Registry* at <http://www.openurl.info/registry/>.

6.2 Registry Identifiers

Upon registration, each instance of a core component receives a unique *Registry Identifier*, which is a URI of the form **info:ofi/char-string**, where:

- **info** is the name of the URI scheme;
- **ofi** represents the namespace under the **info scheme** reserved for *Registry Identifiers*; and
- **char-string** *must* be replaced by a unique character string assigned by the *Registry* upon registration of the instance of the core component.

Table 2 summarizes where to find information related to *Registry Identifiers* for core components. The first column lists core components. The second column displays the structure of their *Registry Identifiers*. The third column lists which Section defines each core component. The fourth and fifth columns (with KEV and XML heading, respectively) give the Sections and Appendices where instances of core components are introduced. This Standard initializes the *Registry* with entries that bootstrap two *Applications* for the scholarly-information community: one *Application* based on the *KEV ContextObject Format* and one based on the *XML ContextObject Format*. These entries *may* also be used by other communities.

Table 2: Core Components and their Registry Identifiers

Core Component	Registry Identifier Structure	Framework	KEV	XML
<i>Serializations</i>	info:ofi/fmt: __	7.1	12.1	16.1
<i>Constraint Languages</i>	info:ofi/fmt: _:__	7.2	12.2	16.2
<i>Character Encodings</i>	info:ofi/enc: __	8.1	13.3	17.4
<i>ContextObject Formats</i>	info:ofi/fmt: _:__ ctx ⁽¹⁾	8.2	12.3.1	17.3
<i>Namespaces</i>	info:ofi/nam: __	9.1	C.5	D.5
<i>Metadata Formats</i>	info:ofi/fmt: _:__ ⁽²⁾	9.2	14.2	18.2
<i>Transports</i>	info:ofi/tsp: __	10	20, 21, 22	20, 21
<i>Community Profiles</i>	info:ofi/pro: __	11	15, Appendix C	19, Appendix D

(1) The last component of *Registry Identifiers* for *ContextObject Formats* *must* start with the reserved prefix **ctx**.

(2) The last component of *Registry Identifiers* for *Metadata Formats* *must not* be named with the reserved prefix **ctx**.

6.3 Using the Registry

Given the *Registry Identifier* of a *Registry* entry, it is possible to obtain the Dublin Core metadata description and the actual definition of the entry. (In the URIs shown below, replace the bold and underlined keyword **registry-identifier** with the *Registry Identifier* of the *Registry* entry.)

The Dublin Core metadata description [18] of the entry is available in two forms:

- for display in a web browser: <http://www.openurl.info/registry/dc/registry-identifier>
- for direct access: <http://www.openurl.info/registry/docs/dc/registry-identifier>

The Dublin Core metadata *may* include **dc:identifier** fields, each containing a URI that points to a definition of the *Registry* entry. This mechanism provides access to multiple forms of the definition.

The following standard form URI always resolves to a definition of the registered resource:

- <http://www.openurl.info/registry/docs/registry-identifier>

If there are one or more **dc:identifier** fields in the *Registry* entry, the standard form URI resolves to the URI contained in the first **dc:identifier** field. There is no standard form URI available to access definitions pointed to by URIs in subsequent **dc:identifier** fields.

In Example 6, the *Registry Identifier* of the DOI Namespace is **info:ofi/nam:info:doi**, and its Dublin Core metadata description is available in two forms:

- for display in a web browser: <http://www.openurl.info/registry/dc/info:ofi/nam:info:doi>
- for direct access: <http://www.openurl.info/registry/docs/dc/info:ofi/nam:info:doi>

The standard form URI <http://www.openurl.info/registry/docs/info:ofi/nam:info:doi> refers to the definition of the registered resource itself (the DOI Namespace), which is described by the Dublin Core metadata shown above.

Because there are one or more **dc:identifier** fields in this *Registry* entry, a resolution mechanism redirects the standard form URI to the URI in the first **dc:identifier** field. This URI, <http://errol.oclc.org/info-uri.info/info:doi/?metadataPrefix=reg>, resolves to a browser display of the definition of the DOI Namespace under the “info” URI scheme.

The URI in the second **dc:identifier** field, <http://errol.oclc.org/info-uri.info/info:doi/.reg>, points to the raw XML record defining the DOI Namespace under the “info” URI scheme (as opposed to the HTML rendition of this record).

The URI in the third **dc:identifier** field points to the info:doi/ namespace in the “info” URI scheme.

In the initial *Registry*, URIs following the pattern <http://www.openurl.info/registry/docs/registry-identifier> are reserved for “native forms”, while the rest of the *Registry* is suitable for web browsing. The initial *Registry* contains the following entry types stored inside the *Registry*:

- Items described by Dublin Core metadata formatted for web browsing use the URI pattern <http://www.openurl.info/registry/dc/registry-identifier>. These same items in their native form use the URI pattern <http://www.openurl.info/registry/docs/dc/registry-identifier>. In this case, the native form is an *XML Document* that conforms to the XML Schema located at http://www.openarchives.org/OAI/2.0/oai_dc.xsd.
- In Section 11, an XML Schema to define *Community Profiles* will be introduced. Web-browsable definitions of *Community Profiles* will use the URI pattern <http://www.openurl.info/registry/pro/registry-identifier>, and XML-based definitions will use the URI pattern <http://www.openurl.info/registry/docs/pro/registry-identifier>.
- In Part 2, Z39.88-2004 Matrix *Constraint Definitions* for *Registry* entries related to the *KEV ContextObject Format* will use the URI patterns <http://www.openurl.info/registry/mtx/registry-identifier> and <http://www.openurl.info/registry/docs/mtx/registry-identifier>.
- In Part 3, XML Schema *Constraint Definitions* for *Registry* entries related to the *XML ContextObject Format* will use the URI pattern <http://www.openurl.info/registry/xsd/registry-identifier> for web-browsable displays and the URI pattern <http://www.openurl.info/registry/docs/xsd/registry-identifier> for XML Schemas.

The initial *Registry* supports the Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH) [17] as a machine interface for downloading *Registry* materials.

7 Formats

To enable the use of a wide variety of *Representations* of *ContextObjects* and their *Entities*, this Standard defines the notion of a *Format*.

A *Format* is a method to represent information constructs as character strings.

Each *Format* consists of a *Serialization*, a *Constraint Language*, and a *Constraint Definition* expressed using the *Constraint Language*. In this Standard, the set of three items defining a *Format* is called a triple and is represented by a short-hand notation as in:

{ *Serialization*, *Constraint Language*, *Constraint Definition* }

In Section 8.2, the *Format* notion is used to define *ContextObject Format*, which gives communities the ability to define, register, and use *ContextObject Representations* that are the most appropriate for their application domain. In Section 9.2, the *Format* notion is used to define *Metadata Format*, which gives communities the ability to define, register, and use appropriate methods to describe *Entities* of *ContextObjects* by means of *By-Value* or *By-Reference Metadata Descriptors*.

7.1 Serializations [Registry]

For representing *ContextObjects* and their *Entities*, this Standard supports the use of a variety of *Serializations*.

A *Serialization* is a method by which structured information can be held in storage and/or can be transmitted over a network.

The description of a resource, such as a *ContextObject* or an *Entity*, is often a hierarchical and complex structure at the conceptual level. The form in which it is stored and/or transmitted over a network, however, is a simple character string. An example of such a storage and/or transmission form is XML.

Serializations **must** be registered before use in an *Application*.

Communities **may** use *Serializations* that are already in the *Registry*, or they **may** register additional *Serializations* as needed.

Upon registration, a *Serialization* is assigned a *Registry Identifier*, formed by concatenating three character strings:

- **info:ofil**, which represents the namespace under the **info** scheme reserved for *Registry Identifiers*
- **fmt:**, the character string used to introduce *Format*-related *Identifiers* in the **info:ofil** *Namespace*
- a character string that is assigned on registration and identifies the *Serialization*.

Registry Identifiers of *Serializations* are used to support *Registry* management and to identify *Serializations* in *Community Profiles*. In typical use, *Registry Identifiers* of *Serializations* do not show up in *Representations* of *ContextObjects* or their *Entities*.

Table 3: Registry Identifiers for Serializations

"info" URI Namespace	Format-related	Serialization	Registry Identifier
info:ofil/	fmt:	kev	info:ofil/fmt:kev
info:ofil/	fmt:	xml	info:ofil/fmt:xml

Table 3 illustrates the construction of *Registry Identifiers* for the two *Serializations* in the initial *Registry*:

- **KEV**: A resource is represented as a string of ampersand-delimited pairs, each pair consisting of a key and an associated URL-encoded value. In the remainder of this Standard, Key/Encoded-Value is abbreviated as KEV. (See Section 12.1.)