ANSI/NISO Z39.88-2003 ISSN: XXXX-XXXX

The OpenURL Framework for Context-Sensitive Services

Status: For Ballot: XXXX, 2003

Abstract: This Standard provides a methodology for describing resources that are referenced in a networked environment as well as for describing the context of the reference. To capture this context, the Standard introduces the concept of the ContextObject and provides a framework for the cross-domain description of such contextual references based on the ContextObject concept. This Standard assumes that representations of ContextObjects describing referenced resources will be transported over networks for the purpose of requesting context-sensitive services pertaining to those referenced resources. This Standard introduces a Registry to contain properties that are fundamental to create concrete representations of ContextObjects, and methods to transport them. It also defines and registers the initial content of that Registry, as a means to bootstrap the deployment of the OpenURL Framework.

A Draft American National Standard
Developed by the
National Information Standards Organization

Published by the National Information Standards Organization
Bethesda, Maryland

NISO Press, Bethesda, Maryland, U.S.A.

DRAFT STANDARD FOR TRIAL USE
SUBJECT TO CHANGE

Published by
NISO Press
4733 Bethesda Avenue, Suite 300
Bethesda, MD 20814
www.niso.org

Printed in the United States of America
ISSN: xxxx
ISBN: xxxx

8 This paper meets the requirements of ANSI/NISO Z39.48-1992 (Rainbow 3D 1997) Permanence of Paper.

Library of Congress Cataloging-in-Publication Data

National Information Standard Organization (U.S.)
xxxx

DRAFT STANDARD SUBJECT TO CHANGE

About NISO Standards

NISO Standards are developed by the Standards 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.

The use of an ANSI/NISO Standard is voluntary. This is, the existence of this NISO Standard does not preclude anyone, whether or not that person has adopted the NISO Standard, from manufacturing, marketing, purchasing, or using products, processes, or procedures that do not conform to the NISO Standard. However, the use of the standards (those developed by NISO as well as other standards-developing organizations) has proven to be in the best interest of any industry wishing to increase its effectiveness and efficiency in the areas of product development, manufacturing, and marketing and, therefore, such use is encouraged by ANSI, NISO, and all other standards-developing organizations.

Contents

Foreword

(This foreword is not part of the American National Standard for the OpenURL Framework, ANSI/NISO Z39.88-2003. It is included for information only.)

NISO Standard Committee AX, OpenURL Standard Syntax, which was organized in 2001, prepared this Standard. The Committee's charge was to develop a mechanism for the representation and transportation of specific packages of metadata and identifiers. Those metadata packages describe a resource that is referenced on a network and the context in which the reference to the resource is made.

The purpose of the representation and transportation of those packages is to facilitate the delivery of services pertaining to the referenced resource. Conventional URL-based linking is inadequate in this respect, because link resolution:

- Is independent of the identity of the agent that actuates the link
- Returns at most one resource or service, not a set of resources and services that depend on the context in which the link is provided and followed
- Fails when linked resources move or become unavailable

The approach taken by the Committee overcomes these limitations by transporting packages of contextual metadata and identifiers to networked systems, named Resolvers. Using the reference and the contextual information contained in these packages, Resolvers may deliver a number of services that pertain to the referenced resource and are appropriate within the context of use.

The Committee's charge emphasized the need for extensibility of the description and transportation mechanisms that were to be devised. Meeting that challenge led the Committee to develop the OpenURL Framework Standard in two Parts:

- "The OpenURL Framework for Context-Sensitive Services, Part 1: ContextObject and Transport Mechanisms", which defines the general framework to bundle specific packages of contextual metadata and transport them over the network. It introduces the Registry to hold core properties of concrete instantiations of the general framework.
- "The OpenURL Framework for Context-Sensitive Services, Part 2: Initial Registry Content", which details core properties that can be used in actual instantiations of the general framework. This Part can be used by some Communities to implement an instantiation of the OpenURL Framework. Other Communities may use this Part as a guideline for the definition and implementation of other instantiations.

Part 1 of this Standard introduces the ContextObject, an information construct containing descriptions of a resource that is referenced on the network and descriptions of resources that form the context of the reference. It also defines a framework consisting of several components named Namespaces (URI, ORI, XRI), Character Encodings, Physical Representations, Constraint Languages, ContextObject Formats, Metadata Formats, Transports, and Community Profiles. When defining an actual instantiation of the general OpenURL Framework, choices for each of these components must be detailed. The Registry is introduced to contain these details.

Part 2 of this Standard lists the initial content of the Registry, and – wherever necessary – also provides detailed definitions of the registered content. The initial Registry content is provided to bootstrap the deployment of two concrete instantiations of the OpenURL Framework, one built upon a Key/Encoded-Value representation for ContextObjects, the

other upon an XML representation for ContextObjects. The initial Registry also details the OpenURL, a suite of HTTP(S)-based methods to transport representations of ContextObjects. The Registry supports the addition of public definitions for the components underlying the general OpenURL Framework, without needing to update this Standard frequently. Communities interested in deploying instantiations of the general OpenURL Framework other than those specified in Part 2, can do so by selecting existing Registry entries, and/or by registering new definitions. These Communities can use Part 2 as a guideline to deploy their instantiation of the OpenURL Framework.

NISO AX Committee

Standards Committee AX had the following members at the time this document was released for Public Comment:

Ms. Ann Apps, The University of Manchester, UK

Mr. Oren Beit-Arie, Ex Libris (USA), Inc.

Mr. Karim Boughida, Getty Research Institute

Ms. Karen Coyle, University of California

Mr. Todd Fegan, ProQuest Information and Learning

Mr. Tony Hammond, Elsevier

Dr. Eric Hellman, Openly Informatics, Inc.

Ms. Lou Knecht, National Library of Medicine

Mr. Larry Lannom, Corporation for National Research Initiatives

Mr. Cliff Morgan, John Wiley & Sons, Ltd.

Mr. Mark Needleman, SIRSI Corporation

Mr. Eamonn Neylon, Manifest Solutions

Mr. Phil Norman, OCLC, Inc.

Mr. Oliver Pesch, EBSCO Publishing

Mr. Harry Samuels, Endeavor Information Systems, Inc.

Dr. Herbert Van de Sompel, Los Alamos National Laboratory

Dr. Eric F. Van de Velde, California Institute of Technology (Committee Chair)

Acknowledgements

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

The Committee thanks the following individuals for their substantial assistance to the process of creating the 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; Justin Littman, netLibrary, Inc.; Pat Stevens, OCLC; 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; 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.

History

Starting in 1999, NISO held a series of invitational workshops to bring together representatives from the library, publishing, and information services communities to explore issues in the area of reference linking. A major issue that emerged from the workshops was the "appropriate copy problem". This 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. Conventional URL-based links cannot accomplish this. The workshop participants recognized that solving the "appropriate copy problem" would lead to solutions for other context-based link resolution problems.

Around the same time, Herbert Van de Sompel (Ghent University, Los Alamos National Laboratory, Cornell University), Patrick Hochstenbach (Ghent University) and Oren Beit-Arie (Ex Libris) worked on a series of efforts aimed at solving the problem of delivering context-sensitive service-links in the web-based scholarly information environment. This is a super-set of the "appropriate copy problem". The result of these efforts was the:

- Development of the SFX linking server and the OpenURL architecture [R1][R2][R3].
- Publication of the OpenURL draft specification, which outlines a URI-based syntax for transporting metadata and identifiers that describe a referenced item along with some contextual information. The transportation is from an information service towards a linking server [R4].
- Publication of the OpenURL Framework, which provides an architecture for context-sensitive reference linking in the Web-based scholarly information environment [R5].

The OpenURL draft specification quickly gained acceptance in the scholarly information industry. Publishers, vendors of abstracting and indexing databases, preprint systems, and the DOI-based CrossRef linking effort introduced OpenURLs in their systems. Many libraries implemented OpenURL conformant linking servers that provided their user base with context-sensitive links.

Experience with using the OpenURL draft specification, and the initiation of its NISO Standardization, led Herbert Van de Sompel and Oren Beit-Arie to introduce the Bison-Futé model. This is a generalization of the OpenURL Framework that identifies components required to enable context-sensitive linking on the Web in general [R6]. The Bison-Futé model introduces the notion of a ContextObject, an information construct that is a generalization, but at the same time a direct derivative, of the contextual information

available in an HTTP GET that conforms to the draft OpenURL specification:

- The draft OpenURL explicitly includes: a referenced resource; the system that provides the OpenURL (sid); and the linking server that is the target of the OpenURL. The Referent, Referrer, and Resolver Entities in the ContextObject are generalizations of these resources, respectively.
- Experience shows that three more resources are regularly described in actual OpenURLs, using the Private Identifier (pid). These are: the initiator of OpenURL transport (the user clicking the OpenURL); the citing scholarly work; and the type of service requested, for example “provide full-text”. The Requester, ReferringEntity, and ServiceType Entities in the ContextObject are generalizations of these types of resources, respectively.
- A referenced scholarly work may be described by inline metadata, using metadata keys that are hardwired into the OpenURL draft specification. A By-Value Metadata Descriptor of Entities in the ContextObject is a generalization of this description method. Metadata Formats are not hardwired into the ContextObject specification, but can be plugged-in in a variety of ways.
- A referenced scholarly work may be described by an identifier. The namespace of this identifier is hardwired into the OpenURL draft. Similarly identifiers may describe the system that provides the OpenURL and the target of the OpenURL. This description method is generalized to an Identifier Descriptor of Entities in the ContextObject. Namespaces of Identifiers are not hardwired into the ContextObject specification, but can be specified in a variety of ways.
- A referenced scholarly work may be described by private data. The syntax of this private data is specific to the system providing the OpenURL. A linking server can understand that syntax only through some prior agreement with the provider of the OpenURL. This method to describe resources becomes a Private Data Descriptor of Entities in the ContextObject.
- Experience shows that the Private Identifier (pid) of the OpenURL draft is commonly used to contain a pointer to metadata describing the referenced scholarly work. This has been generalized into a By-Reference Metadata Descriptor of Entities in the ContextObject.

The above shows that the ContextObject concept has its roots in the draft OpenURL specification. It takes forward lessons learnt from the actual use of OpenURLs in the scholarly information industry. The Committee hopes that the ContextObject specified in this Standard will provide a generic component for systems providing contextual services pertaining to resources that are referenced on the network.

References

[R1] Van de Sompel H, Hochstenbach P. Reference Linking in a Hybrid Libary Environment, Part 1: Frameworks for Linking. D-Lib Magazine, 1999. 5(4). doi:10.1045/april99-van_de_sompel-pt1 [online] [cited 24 February 2003] 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 24 February 2003] 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 24 February 2003] 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 24 February 2003] Available from World Wide Web: http://www.sfxit.com/openurl/openurl.html

[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 24 February 2003] 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 24 February 2003] Available from World Wide Web:http://www.dlib.org/dlib/july01/vandesompel/07vandesompel.html

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 is neutral with respect to applications and user communities. All community specific elements are maintained in a Registry. Initially, this Registry contains elements from the scholarly information community, where the OpenURL draft originated. User communities may add new elements to support different applications. The Registry records the following:

- Formats to express ContextObjects, including their component Physical Representations and Constraint Languages. For example, the XML Format constrained by a W3C XML Schema.
- Namespaces of Identifiers to describe resources
- Metadata Formats to describe particular classes of resources
- Character Encodings
- Methods to transport representations of ContextObjects
- Community Profiles that define choices of the above made by specific communities

Initially the Registry contains two Formats to express ContextObjects: the Key/Encoded-Value (KEV) Format, and the XML Format. The Committee created these to support a wide variety of applications. Further Formats could be defined, following the same rules, and registered at the request of specific communities, thus illustrating the potential of the OpenURL Framework. The initial Registry also contains a suite of HTTP(S)-based methods to transport representations of ContextObjects, named OpenURL. Three Community Profiles are included in the initial content of the Registry. The Committee created these to provide continuing support for the scholarly information community. These three Community Profiles are inspired by the OpenURL draft specification that is already in significant use within the scholarly information Community:

- The San Antonio Profile, level 0: A Community Profile that is introduced to include the OpenURL draft specification into the new OpenURL Framework.
- The San Antonio Profile, level 1: A Community Profile that is based upon the Key-Encoded-Value Format to represent ContextObjects. It uses Identifier 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 backwards compatibility with the OpenURL draft, while at the same time providing enhanced capabilities to

describe referenced resources and the network context in which the references occur.

- The San Antonio Profile, level 2: A Community Profile that is based upon the XML Format to represent ContextObjects. It uses Identifier 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.

ANSI/NISO Z39.88-2003

The OpenURL Framework for Context-Sensitive Services, Part 1: ContextObject and Transport Mechanisms

1. Introduction

1.1 Purpose and Scope

Part 1 of this Standard introduces the ContextObject, an information construct containing descriptions of a resource that is referenced on the network and descriptions of resources that form the context of the reference. The Standard does not constrain the type of resource that can be described in a ContextObject. Part 1 also defines a framework consisting of several components named Namespaces (URI, ORI, XRI), Character Encodings, Physical Representations, Constraint Languages, ContextObject Formats, Metadata Formats, Transports, and Community Profiles. When defining an actual instantiation of the general OpenURL Framework, choices for each of these components must be detailed. The extensible Registry is introduced to contain these details, and Part 2 of this Standard lists the initial Registry content.

Representations of ContextObjects, which contain the contextual description of the referenced resource (the Referent), may reside as autonomous datafiles in information systems. Such static ContextObjects could include descriptions of the referring resource and the system where the reference is made (the ReferringEntity and the Referrer) in addition to a description of the referenced resource.

However, this Standard expects that ContextObjects will be transported between networked systems. Therefore, this Standard introduces the notion of Transports – methods to transport representations of ContextObjects - as a component of the general OpenURL Framework. This Standard does not restrict the purpose of such transportation, but it does anticipate that in many cases it will be a request for context-sensitive services pertaining to the referenced resource that is described in a ContextObject. Therefore, this Standard allows for descriptions of the initiator, target, and purpose of a transportation (the Requester, the Resolver and the ServiceType) to be included within a representation of a ContextObject.

Part 2 of this Standard, introduces specific methods to transport representations of ContextObjects using HTTP(S) GET/POST, named OpenURL. ContextObjects may be transported in a variety of other methods, which Communities may define and register.

1.2. General Structure of Part 1

Part 1 is structured as follows:

- Section 5: The definition of the ContextObject, as an abstract information construct
- Section 6: A description of the Registry
- Sections 7 - 16: The definition of the components of the general OpenURL Framework, i.e. URI Namespaces, ORI Namespaces, Character Encodings, Physical Representations, Constraint Languages, ContextObject Formats, Metadata Formats, Transports, and Community Profiles

2. Related Standards

This Standard relates to or uses the following existing standards:

[1] Extensible Markup Language (XML) 1.0. [online] Second edition. [cited 9 January 2003] Available from World Wide Web : http://www.w3.org/TR/REC-xml

[2] Extensible Markup Language (XML) – XML Path Language (XPATH). [online] [cited 18 February 2003] Available from World Wide Web : http://www.w3.org/TR/xpath

[3] Extensible Markup Language (XML) - XML Schema Part 1: Structures. [online] [cited 9 January 2003] Available from World Wide Web : http://www.w3.org/TR/xmlschema-1/

[4] Extensible Markup Language (XML) - XML Schema Part 2: Datatypes. [online] [cited 9 January 2003] 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. [online] [cited 9 January 2003] Available from World Wide Web: http://www.ietf.org/rfc/rfc2119.txt

[6] IETF RFC 2396, Uniform Resource Identifiers (URI): Generic Syntax. [online] [cited 9 January 2003] Available from World Wide Web: http://www.ietf.org/rfc/rfc2396.txt

[7] Internet Assigned Naming Authority (IANA) List of Registered Character Sets. [online] [cited 9 January 2003] Available from World Wide Web: http://www.iana.org/assignments/character-sets

[8] Internet Assigned Naming Authority (IANA) Uniform Resource Identifier (URI) Schemes. [online] [cited 13 January 2003] Available from World Wide Web: http://www.iana.org/assignments/uri-schemes

[9] Internet Assigned Naming Authority (IANA) URN Namespaces. [online] [cited 13 January 2003] Available from World Wide Web: http://www.iana.org/assignments/urn-namespaces

[10] ISO 8601:2000, Data elements and interchange formats — Information interchange — Representation of dates and times. Available from: International Organization for Standardization, Switzerland.

[11] The Unicode Standard Version 3.0. The Unicode Consortium. Published by Addison-Wesley, Reading, MA. 2000 January. ISBN: 0-201-61633-5. Updated by Unicode 3.2 [online] [cited 9 January 2003] Available from World Wide Web: http://www.unicode.org/unicode/reports/tr28/

[12] W3C Date and Time Formats. [online] [cited 21 February 2003] Available from World Wide Web: http://www.w3.org/TR/NOTE-datetime

3. Notational Conventions

The key words “must”, “must not”, “required”, “shall”, “shall not”, “should”, “should not”, “recommended”, “may”, and “optional” in this Standard are to be interpreted as described in IETF RFC2119. Whenever these key words appear in this Standard in the meaning as specified by IETF RFC2119, they will be shown using a **bold font style**, e.g. **must**, **must not**.

Terms defined in this Standard will be shown in the `Courier New font`, whenever the Standard uses them in the defined meaning, e.g. `ContextObject`, `By-Value Metadata.`

Throughout this document, 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`.

Within the main body of the document, because no formal definitions of specific `ContextObject Formats` are available when key terms are defined, illustrative examples use a simple ‘property list’ syntax. This lists each property, consisting of a key term and associated value, on a separate line. Comment lines begin with a ‘#’ character. The informal syntax for a single property is of the form:

```
<key> = <value>
```

Two `ContextObject Formats` are specified in Part 2 of this Standard. The examples used in those definitions use the syntax of the respective `ContextObject Format` that is being defined.

In tables within this document which specify constraints the following shorthand is used:

Constraint	Minimum Occurrence	Maximum Occurrence
1	1	1
≤ 1	0	1
≥ 0	0	unbounded

Whenever core properties of the OpenURL Framework are defined, of which concrete instantiations must be registered, the title of the Section defining the property has a suffix “[Registry]”

4. Glossary

By-Reference Metadata	A Descriptor that details properties of an Entity by the combination of: (1) a reference to a Metadata Format; and (2) the network location of a particular instance of metadata about the Entity, the metadata being expressed according to the indicated Metadata Format
By-Reference OpenURL	A Transport which uses either the HTTP or HTTPS network protocols for conveying a reference to representations of ContextObjects over a network. The reference to a representation occurs as the Value of a single Key within a Key/Value querystring, which is transported using either an HTTP GET or POST method.
By-Value Metadata	A Descriptor that details properties of an Entity by the combination of: (1) a reference to a Metadata Format; and (2) a particular instance of metadata about the Entity, expressed according to the indicated Metadata Format
By-Value OpenURL	A Transport which uses either the HTTP or HTTPS network protocols for conveying representations of ContextObjects over a network. A representation occurs as the Value of a single Key within a Key/Value querystring, which is transported using either a GET or POST method.
Character Encoding	The combination of a character set and a character encoding for the character set
Constraint Language	A formalism used to specify syntactic and semantic restrictions on information constructs of a given class.
ContextObject	An information construct that binds a description of a primary Entity - the referenced resource - together with descriptions of Entities that indicate the context of the reference to the referenced resource
ContextObject Format	A Format to represent ContextObjects
Community Profile	A consistent selection from among the registered properties of the OpenURL Framework to support a specific application domain
Descriptor	A Descriptor details information about an Entity, using one of the following four methods: Identifier, By-Reference

	Metadata, By-Value Metadata, Private Data
Entity	One of the six possible constituents of a ContextObject: Referent, Requester, Referrer, Resolver, ReferringEntity, or ServiceType
Format	A concrete method of expression for a class of information constructs. It is a triple comprising: (1) a Physical Representation; (2) a Constraint Language; and (3) a set of constraints expressed in that Constraint Language
Identifier	A Descriptor that unambiguously defines an Entity by the combination of: (1) a reference to a Namespace; and (2) a value within that Namespace
Inline OpenURL	A Transport which uses either the HTTP or HTTPS network protocols for conveying representations of ContextObjects over a network. A representation occurs as multiple Key/Value pairs within a Key/Value querystring, which is transported using either a GET or POST method.
Key/Encoded-Value Format	A ContextObject Format to represent a single ContextObject as a string of ampersand-delimited pairs, each pair consisting of a label (Key), and an associated Value that is URL encoded.
KEV Format	See Key/Encoded-Value Format
Metadata Format	A Format to represent an Entity Descriptor by means of By-Reference Metadata or By-Value Metadata
Namespace	The set of all names in a naming system
Naming Environment	A namespace of namespaces
OpenURL	A suite of HTTP(S)-based methods to transport representations of ContextObjects. See also By-Reference OpenURL, By-Value OpenURL, Inline OpenURL
ORI Naming Environment	The Open Resource Identifier Naming Environment (ORI) contains namespaces that are recorded in the Registry because they need to be globally understood, yet are not available in the URI Naming Environment
Physical Representation	A Representation provides a method to hold in storage, or transmit over a network, the values within an information construct

Private Data	A Descriptor that details information about an Entity using a method not defined in this Standard
Referent	The Entity about which the ContextObject was created - a referenced resource
Referrer	The Entity that generated the ContextObject
ReferringEntity	The Entity that references the Referent
Registry	The Registry provides a mechanism to record, and make available publicly, details of the components of the OpenURL Framework: URI Namespaces, ORI Namespaces, Character Encodings, Physical Representations, Constraint Languages, ContextObject Formats, Metadata Formats, Transports, and Community Profiles
Representation	See Physical Representation
Requester	The Entity that requests services pertaining to the Referent
Resolver	The Entity at which a request for services is targeted
ServiceType	The Entity that defines the type of service requested
Transport	A Transport is a network protocol and the method in which it is used to convey representations of ContextObjects
URI Naming Environment	The Uniform Resource Identifier Naming Environment (URI) is the Naming Environment of the Internet Assigned Naming Authority (IANA) consisting of URI Schemes and URN Namespaces
XML Format	A ContextObject Format to represent one or more ContextObjects as an XML instance document
XRI Naming Environment	The Private Resource Identifier Naming Environment (XRI) is the Naming Environment that is local to the Referrer

5. ContextObject, Entity, Descriptor

This Section defines `ContextObject`, `Entity`, and `Descriptor`. They form the basis of an extensible framework for representing references to resources together with the context in which the references occur.

To illustrate the definitions of `ContextObject`, `Entity`, and `Descriptor`, the following scenario is used throughout this Section:

> Jane Doe, a University student at Caltech, 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, it is assumed that ScienceDirect® provides a `ContextObject` instance centered on the reference to the Bergelson article. The provision of this `ContextObject` instance might enable Jane to request the full-text of that article from her institutional linking server. The `Entities` present in that `ContextObject`, as well as possible `Descriptors` for those `Entities`, are shown below.

5.1 ContextObject and Entity

A `ContextObject` is an information construct that binds together descriptions of six `Entities`:

- The `ContextObject` has at its core a primary `Entity` called the `Referent`, which is a resource that is referenced in a particular networked context.
- The other five `Entities` of the `ContextObject` are resources that constitute this networked context. These `Entities` are the `ReferringEntity`, the `Requester`, the `ServiceType`, the `Resolver` and the `Referrer`.

A definition of all `Entities` of the `ContextObjects` is provided in Table 5.1.

Table 5.1: Entities

Entity	Definition	Example
Referent	The `Entity` about which the `ContextObject` was created – a referenced resource	The scholarly article by Bergelson is the `Referent` in the `ContextObject`
Referring Entity	The `Entity` that references the `Referent`	The scholarly article by McArthur is the `ReferringEntity` in the `ContextObject`
Requester	The `Entity` that requests services pertaining to the `Referent`	Jane Doe is the `Requester` in the `ContextObject`
ServiceType	The `Entity` that defines the type of service requested	The type of service requested is the full-text of the referenced article

Resolver	The `Entity` at which a request for services is targeted	Jane Doe's linking server at Caltech is the `Resolver` in the `ContextObject`
Referrer	The `Entity` that generated the `ContextObject`	Elsevier's ScienceDirect® is the `Referrer` in the `ContextObject`

5.2 Descriptor

A `Descriptor` details information about an `Entity`. There are four types of `Descriptor`: `Identifier`, `By-Value Metadata`, `By-Reference Metadata`, and `Private Data`. They are defined in the remainder of this Section.

5.2.1 Identifier

An `Identifier Descriptor` unambiguously specifies the `Entity` by the combination of: (1) a reference to a namespace; and (2) an identifier within that namespace. The identifier can be associated with the `Entity` itself or with metadata for the `Entity`.

> A Digital Object Identifier (10.1126/science.275.5304.1320 in the 'doi' namespace) describes the Bergelson article, which is the `Referent`. Also a PubMed identifier (9036860 in the 'pmid' namespace) unambiguously describes metadata for the Bergelson article.
>
> An e-mail address (jane.doe@caltech.edu in the 'mailto' namespace) can describe Jane Doe, who is the `Requester`.
>
> A URI (http://links.caltech.edu/menu in the 'http' namespace) can describe the institutional linking server at Caltech, which is the `Resolver`.

5.2.2 By-Value Metadata

A `By-Value Metadata Descriptor` details properties of the `Entity` by the combination of: (1) a reference to a `Metadata Format`; and (2) a particular instance of metadata about the `Entity`, expressed according to that `Metadata Format`.

Example 5.1 shows a `By-Value Metadata Descriptor` for a `Referent`, the Bergelson article.

Example 5.1: By-Value Metadata for a Referent

```
rft_val_fmt = xri:fmt:opl:bnf: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
```

In the above example, the Value of the **`rft_val_fmt`** Key (**`xri:fmt:opl:bnf:journal`**) identifies the (fictional) `Metadata Format` that is used to describe the Bergelson article. Sections 11 through 14 explain the meaning and construction of the colon-separated Identifiers for `Formats`.

The remaining lines indicate metadata properties for the `Referent`. The metadata Keys (**`aulast`**, **`auinit`**, **`date`**, etc.) are from the identified `Metadata Format` for a journal article. The prefix, '**`rft.`**', to the metadata keys indicates that the metadata describes a `Referent`.

5.2.3 By-Reference Metadata

A `By-Reference Metadata Descriptor` details properties of the `Entity` by the combination of: (1) a reference to a `Metadata Format`; and (2) the network location of a particular instance of metadata about the `Entity`, expressed according to that `Metadata Format`.

Jane Doe, who is the `Requester`, can be described as in Example 5.2.

Example 5.2: By-Reference Metadata for a Requester

```
req_ref_fmt = xri:http://www.educause.edu/eduperson/rpr
req_ref = uri:ldap://ldap.caltech.edu:389/janed
```

In the above, the value associated with the **`req_ref`** key is a pointer to (i.e.network location of) Jane Doe's entry in the Caltech LDAP directory server. The value of the **`req_ref_fmt`** key specifies, using a private Identifier, 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` details information about the `Entity` using a method not defined in this Standard. The semantics and syntax used are specific to the `Referrer`. This Standard does not provide any global mechanisms to facilitate the interpretation of `Private Data`. Thus interpretation of `Private Data` by the `Resolver` requires a common understanding between the `Referrer` and the `Resolver`.

5.3 Constraints

The number of occurrences of each `Entity` that can be present in a `ContextObject` is constrained:

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

These constraints are summarized in columns 1 and 2 of Table 5.2. The remaining columns 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. No ordering or priority is defined for multiple `Descriptors`

Table 5.2: ContextObject Constraints

Entities	Number	Descriptors			
		Identifier	By-Value Metadata	By-Reference Metadata	Private Data

Referent	1	≥ 0	≥ 0	≥ 0	≥ 0
ReferringEntity	≤ 1	≥ 0	≥ 0	≥ 0	≥ 0
Requester	≤ 1	≥ 0	≥ 0	≥ 0	≥ 0
ServiceType	≥ 0	≥ 0	≥ 0	≥ 0	≥ 0
Resolver	≥ 0	≥ 0	≥ 0	≥ 0	≥ 0
Referrer	≤ 1	≥ 0	≥ 0	≥ 0	≥ 0

In actual representations of the `ContextObject` further constraints **may** apply to the number of occurrences of each `Entity`, as well as to the `Descriptors` that can be used to describe each `Entity`. Those further constraints **must** be specified in `ContextObject Format` definitions. Actual representations of `ContextObjects` **must not** relax the constraints expressed in Table 5.2, i.e. `ContextObject Formats` **must not** specify relaxations of the listed constraints, such as allowing multiple `Referents`, adding `Entities`, adding `Descriptors`, etc.

`ServiceType` and `Resolver` are the only `Entities` that **may** occur more than once in this `ContextObject` data model specified by Table 5.2. Where actual `ContextObject Formats` allow multiple occurrences of these `Entities` they **must** define how multiple `Descriptors` are grouped to bind to particular `Entities`.

6. Registry

The OpenURL Framework builds on several components that are introduced to support the cross-domain description of network-referenced resources, the description of the context of the references, and the transportation of those descriptions over the network. Those components are introduced in the remainder of Part 1. They are Namespaces (URI, ORI, XRI), Character Encodings, Physical Representations, Constraint Languages, ContextObject Formats, Metadata Formats, Transports, and Community Profiles.

When Communities define an actual instantiation of the general OpenURL Framework, choices for each of these components must be detailed. This Standard assumes the existence of a Registry to contain these details, and Part 2 shows the initial content of the Registry. It contains many entries that can be used by several Communities, and some that are specific for the scholarly information community, where the OpenURL draft originated.

The Registry is based at http://www.openurl.info/ . When details for the components of the OpenURL Framework are registered, they receive a unique identifier in the Registry. Identifiers for these components have a prefix ‘uri:’ or ‘ori:’ , and the meaning of these prefixes will be explained in Section 7.

The content of the Registry is read-only for the Trial Use of the Standard. After Trial Use, it will be possible to register new elements to support particular applications and Communities. Further details about the Registry including the process and procedures for adding items to it will be provided in a separate document.

Table 6.1 shows the components of the OpenURL Framework, the manner in which they are identified in the Registry, the Section in Part 1 in which their general properties are defined, and the Section in Part 2 that details specific choices. Figure 6.1. shows these components in a graphic manner, and clarifies how Community Profiles list Community-specific choices for the general components of the OpenURL Framework.

Table 6.1: Components of OpenURL Framework and their identification

OpenURL Framework component	Identifier structure	Section in Part 1	Section in Part 2
URI Namespaces	uri:_	7	2
ORI Namespaces	ori:_	7	3
XRI Namespaces	xri:_	7	-
Character Encodings	ori:enc:_	8	4
Physical Representations	ori:fmt:_	9	5
Constraint Languages	ori:fmt:_:_	10	6
ContextObject Formats	ori:fmt:_:_:ctx	12	7, 8
Metadata Formats	ori:fmt:_:_:_	13	9, 10
Transports	ori:tsp:_	14	11
Community Profiles	ori:pro:_	15	12

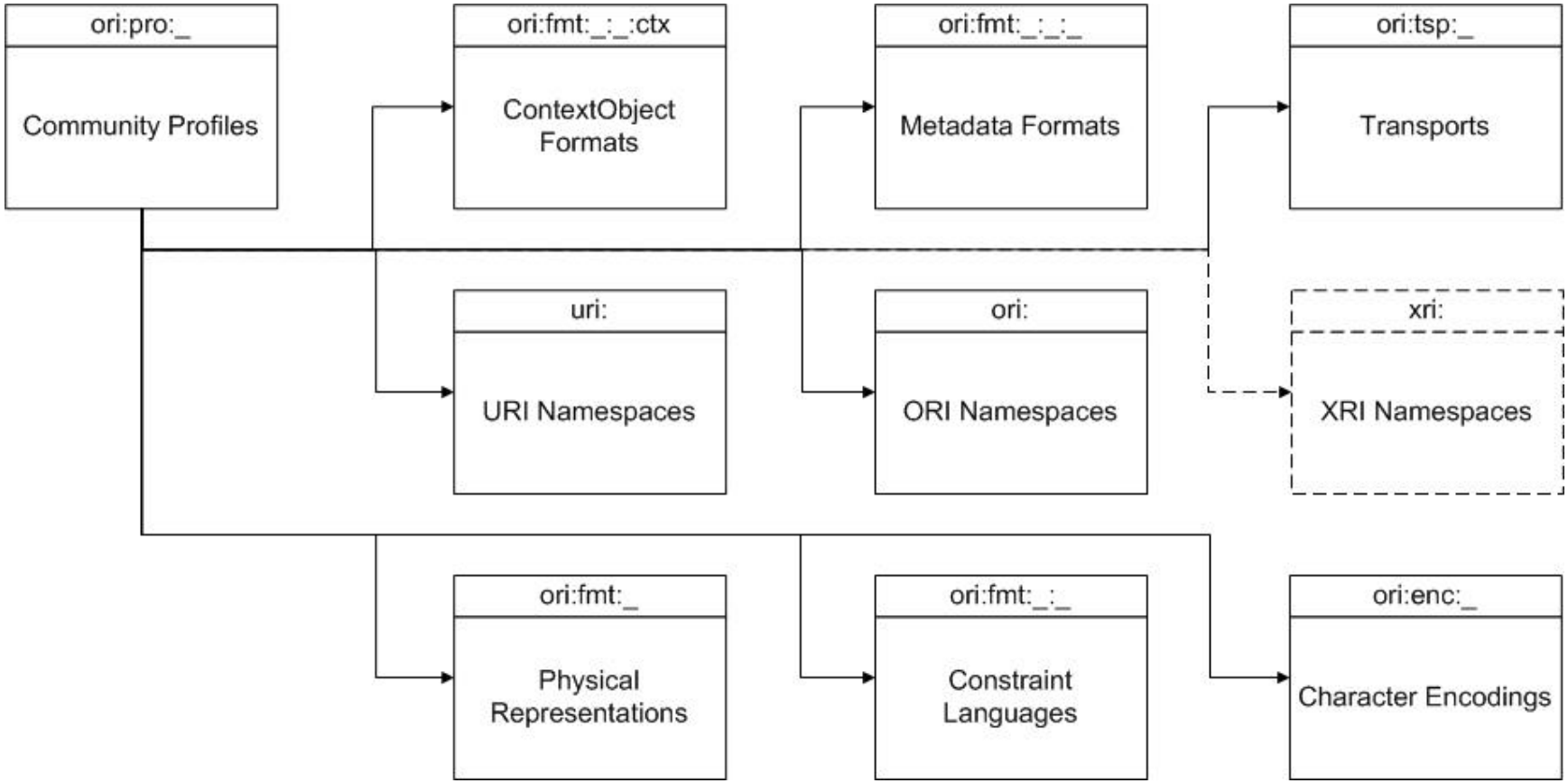

Figure 6.1: Components of the OpenURL Framework

7. Naming Environments for Identifiers [Registry]

To allow extensible description of Entities by means of Identifiers, this Standard introduces the concept of a Naming Environment.

A Naming Environment is a namespace for namespaces of Identifiers. This Standard introduces three such Naming Environments – URI, ORI, XRI – each serving a specific purpose, as described below.

7.1 The Uniform Resource Identifier (URI) Naming Environment

The Uniform Resource Identifier (URI) Naming Environment is the Naming Environment of the Internet Assigned Naming Authority (IANA). It contains all URI schemes registered with IANA including the registered URN Namespace Identifiers.

Registries on which the URI Naming Environment is based:

- IANA URI scheme Registry [8]: http://www.iana.org/assignments/uri-schemes
- IANA URN Namespace Identifiers Registry [9]: http://www.iana.org/assignments/urn-namespaces

These registries are an essential part of the Internet infrastructure, and therefore URIs and URNs can be understood globally.

To support the usage of URI schemes and URN Namespaces for the identification of Entities in the OpenURL Framework, this Standard requires them to be recorded in the Registry. URI schemes and URN Namespaces registered in the Registry are referred to as URI Namespaces. Identifiers from URI Namespaces are qualified with the reserved string 'uri'.

URI Namespaces that can be used for the identification of Entities by means of Identifier Descriptors **must** be registered. Valid URI Namespaces **must** be taken from the Internet Assigned Naming Authority (IANA) Lists [8] [9]. When an Entity is described by means of an Identifier from a registered URI Namespace, that description **must** follow the specifications of the IETF RFCs of the corresponding IANA URI Scheme, as shown in the above IANA registries.

Part 2 lists the URI Namespaces that form the initial content of the Registry. Through the registration mechanism, URI Namespaces **may** be added to the Registry. This Standard does not support the use of unregistered URI Namespaces.

Example 7.1 shows Identifiers from URI Namespaces used for the identification of a Referent, Requester, and Referrer.

Example 7.1: Identifying Entities using URI Namespaces

```
rft_id = uri:urn:ISBN:0262011808
req_id = uri:mailto:jane.doe@caltech.edu
rfr_id = uri:http://www.sciencedirect.com
```

7.1.1. Identifiers for URI Namespaces

URI Namespaces **must** be registered, and upon registration they obtain an Identifier in the Registry. The structure of these Identifiers **must** be as specified in the remainder of this Section.

Table 7.1 shows the components used to construct Identifiers for URI Namespaces. These components represent:

- The Naming Environment of the Identifier, which **must** be 'uri'

- A string indicating the actual URI Scheme or URN Namespace followed by a colon:
  - For URI Schemes, the string to be used is the one listed in the Column "Scheme Name" at [8]
  - For URN Namespaces, the string to be used is the one listed under the Column "Registered Formal URN Namespaces" at [9]. This string **must** be preceded by 'urn:'.

Table 7.1: Construction of Identifiers for URI Namespaces

Naming Environment	URI Scheme or URN Namespace
`uri`	`http`
`uri`	`mailto`
`uri`	`urn:ISSN`

The `Identifier` for a `URI Namespaces` is obtained by concatenating the various components, separated by colons. For example:

- **uri:http** is the `Identifier` of the registered `URI Namespace` that references the http URI Scheme as defined by RFC2616
- **uri:mailto** is the `Identifier` of the registered `URI Namespace` that references the mailto URI Scheme as defined by RFC2368
- **uri:urn:ISSN** is the `Identifier` of the registered `URI Namespace` that references the ISSN URN Namespace as defined by RFC3044

`Identifiers` of `URI Namespaces` are used in actual representations of the `ContextObject`, for the identification of the namespace of an `Identifier Descriptor`, and `Metadata Formats`. They are also introduced to support `Registry` management and unambiguous expression of `Community Profiles`.

7.2 The Open Resource Identifier (ORI) Naming Environment

Many commonly used Community-specific schemes for the identification of resources are not part of the Internet infrastructure. To support such schemes this Standard allows namespaces of such identifiers to be recorded in the `Registry`. The Open Resource Identifier (`ORI`) `Naming Environment` contains namespaces that are recorded in the `Registry`. Such namespaces are referred to as `ORI Namespaces`. `Identifiers` from `ORI Namespaces` are qualified with the reserved string 'ori'.

`ORI Namespaces` that can be used for the identification of `Entities` by means of `Identifier Descriptors` **must** be recorded in The `Registry`. When an `Entity` is described by means of an `Identifier` from a registered `ORI Namespace`, that description **must** follow the specifications of the registered `ORI Namespace` as contained in or linked to by the `Registry`.

The `Registry` for `ORI Namespaces` on which the `ORI Naming Environment` is based is:

- The OpenURL Framework `Registry`: http://www.openurl.info/

This `Registry` is public. Therefore, `Identifiers` in the `ORI Naming Environment` are globally understood. Part 2 of this Standard lists the `ORI Namespaces` that form the

initial content of the Registry. Through the registration mechanism, ORI Namespaces **may** be added to the Registry. This Standard does not support the use of unregistered ORI Namespaces.

Example 7.2 shows Identifiers from ORI Namespaces used for the identification of a Referent, ReferringEntity, and Referrer. The Referent is described using its PubMed identifier in the 'ori:pmid:' ORI Namespace. The ReferringEntity is described by its Digital Object Identifier in the 'ori:doi:' ORI Namespace. The Referrer is a version of the Inspec database provided by OCLC, which is described by an identifier 'oclc.org:Inspec' in the 'ori:rfr:' ORI Namespace. Within the 'ori:rfr:' ORI Namespace, Referrers are identified by a colon-separated combination of their DNS name and the application/resource name. This is explained in Part 2. Example 7.2. also shows how a registered Metadata Format is identified in the 'ori:fmt:' ORI Namespace.

Example 7.2: Identifying Entities using ORI Namespaces

```
#Use of registered ORI Namespaces to identify a

#Referent,#ReferringEntity and Referrer:

rft_id = ori:pmid:9036860
rfe_id = ori:doi:10.1126/science.275.5304.1320
rfr_id = ori:rfr:oclc.org:Inspec

#Use of the registered 'ori:fmt:' ORI Namespace to identify
#a Metadata Format:
rft_val_fmt = ori:fmt:kev:mtx:journal
```

7.2.1. Identifiers for ORI Namespaces

ORI Namespaces **must** be registered, and upon registration they obtain an Identifier in the Registry. The structure of these Identifiers **must** be as specified in the remainder of this Section.

Table 7.2 shows the components used to construct Identifiers for ORI Namespaces. These components represent:

- The Naming Environment of the Identifier, which **must** be 'ori'
- A name assigned on registration to indicate the identifier namespace, followed by a colon

Table 7.2: Construction of Identifiers for ORI Namespaces

Naming Environment	Identifier Namespace
ori	**doi**
ori	**pmid**
ori	**fmt**

The ORI Identifier for a registered namespace of identifiers is obtained by concatenating the two components, separated by colons. For example:

- **ori:doi:** is the Identifier of the registered ORI Namespace consisting of Digital Object Identifiers
- **ori:pmid:** is the Identifier of the registered ORI Namespace consisting of

PubMed Identifiers

- **ori:fmt:** is the `Identifier` of the registered `ORI Namespace` consisting of `Identifiers` for `Formats`

`Identifiers` of `ORI Namespaces` are used in actual representations of the `ContextObject`, for the identification of the namespace of an `Identifier Descriptor, Character Encodings,` and `Metadata Formats`. They are also introduced to support `Registry` management and unambiguous expression of `Community Profiles`.

The following example shows some uses of the `ORI Namespace` in a representation of a `ContextObject`.

7.3 The Private Resource Identifier (XRI) Naming Environment

Some identifier schemes are specific to a given information resource and there may be no need for them to be understood globally. The Private Resource Identifier (`XRI`) `Naming Environment` contains namespaces that are specific to `Referrer` environments.

There are no global mechanisms available to facilitate the interpretation of an `Identifier` namespace in this `Naming Environment`. Thus, interpretation of an `Identifier` in the `XRI Naming Environment` by the `Resolver` requires a common understanding between the `Referrer` and the `Resolver`. Namespaces in the `XRI Naming Environment` are referred to as `XRI Namespaces`. `Identifiers` from `XRI` Namespaces are qualified with the reserved string 'xri'.

Example 7.3 shows `Identifiers` from `XRI Namespaces` used for the identification of a `Referent`, and a `ServiceType`.

Example 7.3: Identifying Entities using XRI Namespaces

```
rft_id = xri:ERL:MX01:123059849
svc_id = xri:my-service:addToCart
```

7.3.1. Identifiers for XRI Namespaces

`XRI Namespaces` are not registered. The structure of `Identifiers` in `XRI Namespaces` **must** be as specified in the remainder of this Section.

Table 7.3 shows the components used to construct ORI `Identifiers`. These components represent:

- The `Naming Environment` of the `Identifier`, which **must** be 'xri'
- A name assigned by the `Referrer` to identify an `Entity`

Table 7.2: Construction of Identifiers in XRI Namespaces

Naming Environment	Identifier Namespace (*)
xri	**-**

(*) Identifiers in `XRI Namespaces` are specific to a `Referrer`-environment

8. Character Encodings [Registry]

Descriptions of resources can be expressed using a variety of Character Sets and Character Encodings. In the remainder of this Section, the combination of Character Sets and Character Encodings are referred to as `Character Encodings`.

`Character Encodings` that can be used for the representation of `ContextObjects` **must** be registered. Valid `Character Encodings` **must** be taken from the Internet Assigned Naming Authority (IANA) List [7]. When a representation of a `ContextObject` indicates the use of a registered `Character Encoding` that description **must** follow the specification of the corresponding IANA character set, as shown in the IANA list.

Part 2, Section 4, defines the `Character Encodings` that form the initial content of the `Registry`. Through the registration mechanism, `Character Encodings` **may** be added to the `Registry`. This Standard does not support the use of unregistered `Character Encodings`.

8.1 Identifiers for Character Encodings

`Character Encodings` **must** be registered with an `Identifier` in the `ORI Naming Environment`. The structure of these `Identifiers` **must** be as specified in the remainder of this Section.

Table 8.1 shows the components used to construct `ORI Identifiers` for `Character Encodings`. These components represent:

- The `Naming Environment` of the `Identifier`, which **must** be 'ori'
- The Namespace dedicated to `Identifiers` of `Character Encodings` in the `ORI Naming Environment`, which **must** be 'enc'
- The IANA character set [7] represented by:
  - The value with an indication of "preferred MIME name", if such a value is available
  - The value of the "Name" tag, if the "preferred MIME name" is not available

Table 8.1: Construction of ORI Identifiers for Character Encodings

Naming Environment	Name-space	IANA Name
ori	**enc**	**UTF-8**
ori	**enc**	**Big5**
ori	**enc**	**ISO-8859-1**

The ORI `Identifier` for a `Character Encoding` is obtained by concatenating the various components, separated by colons. For example:

- **ori:enc:UTF-8** is the `Identifier` of the registered `Character Encoding` that references the IANA character set with the Name "UTF-8"
- **ori:enc:Big5** is the `Identifier` of the registered `Character Encoding` that references the IANA character set with the Name "Big5"
- **ori:enc:ISO-8859-1** is the `Identifier` of the registered `Character Encoding`

that references the IANA character set with the Name “ISO_8859-1:1987”, and with the preferred MIME name “ISO-8859-1”

`Identifiers` of `Character Encodings` will typically be used to indicate the character set and character encoding of a representation of a ContextObject. They are also introduced to support `Registry` management and unambiguous expression of `Community Profiles`.

The following example shows the use of an Identifier of a `Character Encoding` to indicate the character set and character encoding that is used to represent a ContextObject with an administrative key.

Example 8.1: Identification of Character Encoding in a ContextObject

```
ctx_enc = ori:enc:UTF-8
```

9. Physical Representations [Registry]

This Standard defines an extensible framework that allows for the description of a wide variety of resources. Those descriptions can be expressed using a variety of Physical Representations. A Physical Representation provides a method to hold in storage or transmit over a network the values within an information construct. It is the serialization of the information construct into a recognized syntax.

Physical Representations that can be used for the representation of ContextObjects **must** be registered. Through the Registry this Standard initially supports two Physical Representations:

- Key/Encoded-Value (KEV): the information construct is represented as a string of ampersand-delimited pairs, each pair consisting of a name (Key), and an associated Value that is URL encoded
- XML: the information construct is represented as an XML instance document

Part 2, Section 5 of this Standard lists the Physical Representations that form the initial content of the Registry. Through the registration mechanism, further Physical Representations **may** be added to the Registry.

9.1 Identifiers for Physical Representations

Physical Representations **must** be registered with an Identifier in the ORI Naming Environment. The structure of these Identifiers **must** be as specified in the remainder of this Section.

Table 9.1 shows the components used to construct ORI Identifiers for Physical Representations. These components represent:

- The Naming Environment of the Identifier, which **must** be 'ori'
- The Namespace dedicated to Identifiers of Formats in the ORI Naming Environment, which **must** be 'fmt'
- A name assigned on registration to indicate the Physical Representation

Table 9.1: Construction of ORI Identifiers for Physical Representations

Naming Environ-ment	Name-space	Physical Representation
ori	fmt	kev
ori	fmt	xml

The ORI Identifier for a Physical Representation is obtained by concatenating the various components, separated by colons. Thus the ORI Identifier for the KEV Physical Representation is '**ori:fmt:kev**', and that for XML is '**ori:fmt:xml**'.

Identifiers of Physical Representations are used as the leading part of Identifiers of ContextObject Formats and Metadata Formats (see Sections 11 - 14). They are also introduced to support Registry management, and unambiguous expression of Community Profiles.

10. Constraint Languages [Registry]

Constraints on the manner in which resources are described can be expressed by a variety of Constraint Languages. A Constraint Language is used to specify syntactic and semantic restrictions on information constructs of a given class. A Constraint Language is typically bound to a specific Physical Representation of information constructs.

Constraint Languages that can be used for the representation of ContextObjects **must** be registered. Through the Registry, this Standard initially supports two constraint languages:

- Z39.88-2003 MTX: This Standard provides a simple Constraint Language, a Matrix, to specify constraints for a string of ampersand-delimited Key/Encoded-Value pairs in a human readable way. It is listed as a Constraint Language in Part 2, Section 6, and defined in Part 2 Appendix A.
- W3C XML Schema: The W3C XML Schema definition language, endorsed by the World Wide Web Consortium (W3C), may be used to describe the structure and constrain the contents of XML 1.0 documents. This Standard uses it to specify constraints for XML instance documents. It is listed as a Constraint Language in Part 2, Section 6.

Part 2, Section 6, lists the Constraint Languages that form the initial content of the Registry. Through the registration mechanism, further Constraint Languages **may** be added to the Registry.

10.1 Identifiers for Constraint Languages

Constraint Languages **must** be registered with an Identifier in the ORI Naming Environment. The structure of these Identifiers **must** be as specified in the remainder of this Section.

Table 10.1 shows the components used to construct ORI Identifiers for Constraint Languages. These components represent:

- The Naming Environment of the Identifier, which **must** be 'ori'
- The Namespace dedicated to Identifiers of Formats in the ORI Naming Environment, which **must** be 'fmt'
- A name assigned on registration to indicate the Physical Representation
- A name assigned on registration to indicate the Constraint Language

Table 10.1: Construction of ORI Identifiers for Constraint Languages

Naming Environment	Name-space	Physical Representation	Constraint Language
ori	**fmt**	**kev**	**mtx**
ori	**fmt**	**xml**	**xsd**

The ORI Identifier for a Constraint Language is obtained by concatenating the various components, separated by colons. Thus the ORI Identifier for the Z39.88-2003 MTX Constraint Language associated with the Key/Encoded-Value Physical

`Representation` is '**ori:fmt:kev:mtx**', and that for W3C XML Schema `Constraint Language` is '**ori:fmt:xml:xsd**'.

`Identifiers` of `Constraint Languages` are used as the leading part of `Identifiers` of `ContextObject Formats` and `Metadata Formats` (see Sections 11 – 14). They are also introduced to support `Registry` management, and unambiguous expression of `Community Profiles`.

11. Formats

This Section introduces the general concept of Format as used in this Standard. Sections 12 and 13 use this concept to derive specific implementations of Formats for ContextObjects and for Metadata to describe Entities.

A Format is a concrete method to express a class of information constructs. Each Format consists of three items: a Physical Representation, a Constraint Language, and a set of constraints expressed using the Constraint Language. A Format is defined by declaring a choice for each of these three items from those available in the Registry. In this document, the set of three items defining a Format is called a 'triple' and is depicted by a shorthand notation as:

{ Physical Representation, Constraint Language, Set of Constraints }

For example, the Format triple for the initially registered XML ContextObject Format (see Part 2, Section 8) is shown as (where XSD is an abbreviation for the W3C XML Schema Constraint Language):

{ XML, XSD, XML Schema of Part 2 Section 8 }

The Format triple also appears, following 'ori:fmt:' in Identifiers for Formats, as described in Sections 9 and 10. For example, the Identifier for a Metadata Format to describe books using the initially registered XML Format could be '**ori:fmt:xml:xsd:book**'. In the Registry, an XML Schema that defines Metadata Keys, their semantics and constraints would uniquely correspond with this Identifier.

Thus a Format triple consists of a choice for each of the following:

- A **physical representation** for the information construct. Physical Representations are described in Section 9. The two initially registered Physical Representations (Key/Encoded-Value and XML) are listed in Part 2, Section 5.
- A **constraint language** appropriate to that Physical Representation. Constraint Languages are described in Section 10. The two initially registered Constraint Languages (MTX and W3C XML Schema) are listed in Part 2, Section 6.
- A **set of constraints** that apply to the Physical Representations, expressed by means of a Constraint Language. This set of constraints would typically include a semantic description as well as syntactic properties and cardinality constraints. Through the Registry, this Standard initially supports two types of Constraint Language instance documents:
  - Z39.88-2003 MTX Matrices
  - W3C XML Schema documents

Initially, the Registry contains Constraint Language instance documents that define two ContextObject Formats and Metadata Formats to describe specific classes of resources. Other Constraint Language instance documents **may** be added through the Registry mechanism.

12. ContextObject Formats [Registry]

This Standard defines an extensible framework that allows for various methods to represent ContextObjects, known as ContextObject Formats. Sections 7 and 8 of Part 2 of this Standard explicitly defines two such methods, which form the initial content of the Registry. Other methods **may** be added through the Registry mechanism.

12.1. Definition of ContextObject Format

The ContextObject Format specifies the concrete choices made for items within the Format triple, thus declaring the actual representation of the ContextObject. The two initially registered ContextObject Formats are:

- The Key/Encoded-Value(KEV)Format depicted by the triple
  { KEV, MTX, Matrix of Part 2, Section 7 }
  The ContextObject is represented as a string of ampersand-delimited Key/Encoded-Value pairs, constrained using the Z39.88-2003 MTX matrix, by means of the Matrix of Section 7 of Part 2.
- The XML Format depicted by the triple
  { XML, XSD, XML Schema of Part 2, Section 8 }
  The ContextObject is represented as an XML instance document, constrained using the W3C XML Schema language (XSD), by means of the XML Schema of Section 8 of Part 2.

Through the Registry mechanism, other ContextObject Formats **may** be supported. For example, new XML Representations constrained by means of other syntactic constraint languages (such as a DTD, RELAX NG) or semantic constraint languages (such as RDFS, OWL) **may** be introduced for future applications. All definitions of ContextObject Formats **must** rigorously adhere to the Format approach described in the above.

A ContextObject Format **may** further restrict the constraints expressed in Table 5.2. For example, it **may** further restrict the number of Entities and the manner in which Entities can be described. A ContextObject Format **must not** relax the constraints expressed in Table 5.2.

Administrative information **may** be included in ContextObject Formats by the Communities who register them. The addition of administrative items is **recommended** and **may** be required by the Registry. The administrative information included in the KEV and XML ContextObject Formats is defined in Part 2, Sections 7 and 8, respectively.

12.2 Identifiers for ContextObject Formats

ContextObject Formats **must** be registered. The Registry **must** assign an Identifier in the ORI Naming Environment to each registered ContextObject Format. The structure of these Identifiers **must** be as specified in the remainder of this Section.

The construction of Identifiers for ContextObject Formats builds on the Identifiers for Physical Representations and Constraint Languages defined in Sections 9 and 10. Table 12.1 shows the components used to construct ORI Identifiers for ContextObject Formats. These components represent:

- The Naming Environment of the Identifier, which **must** be ‘ori’
- The Namespace dedicated to ContextObject Format Identifiers in the

`ORI Naming Environment`, which **must** be ‘fmt’

- Each entry of the triple that specifies the `ContextObject Format`:
  - The physical `Representation`: through the `Registry`, initially, only ‘kev’ and ‘xml’ are supported by this Standard, but other representations **may** be registered
  - The `Constraint Language`: through the `Registry`, initially, only ‘mtx’ (for Z39.88-2003 MTX) and ‘xsd’ (for W3C XML Schema language) are supported by this Standard.
  - The constraint definition, which **must** be ‘ctx’ to specify that this is a `Format` for a `ContextObject`

Table 12.1: Construction of ORI Identifiers for ContextObject Formats

Naming Environment	Name-space	Repre-sentation	Constraint Language	Constraint Definition
ori	**fmt**	**kev**	**mtx**	**ctx**
ori	**fmt**	**xml**	**xsd**	**ctx**

The ORI `Identifier` for a `ContextObject Format` is obtained by concatenating the various components, separated by colons. For example:

- **ori:fmt:kev:mtx:ctx** identifies the `KEV Format` for the `ContextObject` constrained by a Matrix (see Part 2, Section 7)
- **ori:fmt:xml:xsd:ctx** identifies the `XML Format` for the `ContextObject` constrained by a W3C XML Schema (see Part 2, Section 8)

13. Metadata Formats [Registry]

This Standard defines an extensible framework that allows a wide variety of resources to be described by a wide variety of By-Value Metadata and By-Reference Metadata descriptions. This is achieved through the use of Metadata Formats.

Metadata Formats **may** be registered. Sections 9 and 10 of Part 2 list and/or define the Metadata Formats that form the initial content of the Registry. Through the registration mechanism, Metadata Formats **may** be added to the Registry. This Standard also supports the use of Metadata Formats that are not registered (see Sections 13.2.1 and 13.2.3).

13.1. Definition of Metadata Format

A Metadata Format is a Format to represent an Entity by means of By-Value Metadata or By-Reference Metadata. Being a Format, it specifies the triple of concrete choices { Physical Representation, Constraint Language, set of constraints } made for the actual representation of the By-Value Metadata or By-Reference Metadata.

A variety of Metadata Formats **may** be used to describe Entities of the ContextObject. For example, to describe a class of resources of type ‘book’, Metadata Formats could be specified by the following triples:

- { KEV, MTX, Matrix for type Book (Registry) }. The By-Value Metadata or By-Reference Metadata description is represented as a string of ampersand-delimited Key/Encoded-Value pairs constrained by a registered MTX Matrix.
- { XML, XML Schema, XML Schema for type Book (Registry) }. The By-Value Metadata or By-Reference Metadata description is represented as an XML instance document, constrained by a registered XML Schema.

13.2 Identifiers for Metadata Formats

Metadata Formats **may** be registered with Identifiers in the ORI Naming Environment assigned by the Registry. Metadata Formats **may** also be established in public or private environments, with Identifiers residing in the URI or the XRI Naming Environment, respectively. The following Sections describe the properties of the Metadata Formats in relation to the Naming Environment in which they are defined.

13.2.1 Metadata Format Identification in the URI Naming Environment

In the URI Naming Environment, a URL identifies a Metadata Format. This URL is the network location of the definition of the Metadata Format, which **must** be a document that specifies a set of constraints expressed by means of a registered Constraint Language.

A corresponding By-Value Metadata or By-Reference Metadata description of an Entity of the ContextObject:

- **must** be an instance document that is valid according to the identified Metadata Format.
- **must** be expressed by means of a registered Physical Representation
- **must** be expressed by means of a registered Constraint Language

The following example shows a possible Identifier in the URI Naming Environment for Metadata Formats for a By-Value Metadata description.

Example 13.1: Metadata Format Identification in URI Naming Environment

```
rft_val_fmt = uri:http://www.example.net/mtx/cars.html
```

13.2.2 Metadata Format Identification in the ORI Naming Environment

In the ORI Naming Environment, a Metadata Format is identified by means of an ORI Identifier that **must** be assigned by the Registry. The structure of those Identifiers **must** be as specified in the remainder of this Section. In the Registry, each Identifier **must** uniquely correspond with a definition of the Metadata Format that specifies a set of constraints expressed by means of a registered Constraint Language.

When a Metadata Format is identified in the ORI Naming Environment, a corresponding By-Value Metadata or By-Reference Metadata description of an Entity of the ContextObject:

- **must** be an instance document that is valid according to the definition that, in the Registry, corresponds with the Metadata Format Identifier

The construction of Identifiers for ContextObject Formats builds on the Identifiers for Physical Representations and Constraint Languages defined in Sections 9 and 10. Table 13.1 shows the components used to construct ORI Identifiers for Metadata Formats. These components represent:

- The Naming Environment of the Identifier, which **must** be 'ori'
- The Namespace, which **must** be 'fmt'
- Each entry of the triple that specifies the Metadata Format:
  - The physical Representation: through the Registry, initially, only 'kev' and 'xml' are supported by this Standard, but other representations **may** be registered
  - The Constraint Language: through the Registry, initially, only 'mtx' (for Z39.88-2003 MTX) and 'xsd' (for W3C XML Schema language) are supported by this Standard.
  - The constraint definition: a name assigned to the Metadata Format on registration. In order to suggest relationships between variant Metadata Formats used to describe the same class of Entities, the Registry **may** assign them the same name. This strategy is illustrated in Table 12.1 by using 'book' as the name for Metadata Formats to describe books using KEV and XML representations.

Table 13.1: Construction of ORI Identifiers for Metadata Formats

Naming Environment	Name-space	Represent-ation	Constraint Language	Constraint Definition
ori	**fmt**	**kev**	**mtx**	**book**
ori	**fmt**	**xml**	**xsd**	**book**

The ORI Identifier for a Metadata Format is obtained by concatenating the various components, separated by colons. For example:

- **ori:fmt:kev:mtx:book** could be the ORI Identifier of a Metadata

Format to describe a book using the Key/Encoded-Value representation

- **ori:fmt:xml:xsd:journal** could be the ORI Identifier of a Metadata Format to describe a journal using the XML representation
- **ori:fmt:xml:xsd:marc21** could be the ORI Identifier of a Metadata Format to describe resources using the XML representation using MARC21 XML.

13.2.3 Metadata Format Identification in the XRI Naming Environment

In the XRI Naming Environment, a Referrer-specific Identifier indicates a Metadata Format. No global mechanisms are available to support an understanding of the identified Metadata Format.

This Standard does not define the nature of metadata formats that are identified in the XRI Naming Environment. Such metadata formats are not necessarily Metadata Formats in the sense defined by this Standard, and they may be defined in a manner other than the triple-approach that forms the basis of a Format.

For example:

xri:service_04 could be an XRI Identifier of a Metadata Format to describe a ServiceType

13.3 Independence of Metadata Formats

In order to avoid complexity, this Standard does not specify any relationship between variant Metadata Formats for the By-Value Metadata or By-Reference Metadata description of Entities. Specifically, it does not define a formal relationship between KEV and XML Metadata Formats used to represent the same class of Entities.

14. Format dependencies

The following requirements are introduced to guarantee a minimum level of consistency between `ContextObject Formats` and `Metadata Formats`.

The `Physical Representation` of `By-Value Metadata` descriptions and the `Physical Representation` of their containing `ContextObject` **must** be the same. This requirement is made to support straightforward embedding of `By-Value Metadata` descriptions. It is irrespective of the `Naming Environment` in which the `Metadata Formats` are identified. Thus:

- For a `KEV ContextObject` all `By-Value Metadata` descriptions **must** be expressed as a string of Key/Encoded-Value pairs that conform to an MTX Matrix
- For an XML `ContextObject` all `By-Value Metadata` descriptions **must** use XML as a physical representation, and the XML must conform to an XML Schema

The `Physical Representation` of `By-Reference Metadata` descriptions and the `Physical Representation` of the `ContextObject` referring to them **must** be the same whenever `Metadata Formats` are identified in the `URI Naming Environment`. This requirement guarantees a minimal level of understanding of `Metadata Formats` that are not in the `Registry`, but are identified dynamically instead. Thus:

For a KEV `ContextObject,` if a `Metadata Format` is identified in the `URI Naming Environment`, then the corresponding `By-Reference Metadata` description **must** be expressed as a string of Key/Encoded-Value pairs that conform to the MTX Matrix at the network-location specified by that URI `Identifier`

For an XML `ContextObject,` if a `Metadata Format` is identified in the `URI Naming Environment`, then the corresponding `By-Reference Metadata` description **must** be an XML instance document that conforms to the XML Schema at the network-location specified by that URI `Identifier`.

15. Transports [Registry]

This Standard assumes that representations of `ContextObjects` will be transported over the network. The method of transportation referred to as `Transport` in this Standard. A `Transport` specifies:

- The network protocol that is used to transport representations of `ContextObjects`
- The method in which the network protocol is used

`Transports` that can be used to transport representations of `ContextObjects` **must** be registered.

Part 2, Section 11, lists and defines the `Transports` that form the initial content of the `Registry`. Through the registration mechanism, `Transports` **may** be added to the `Registry`. This Standard does not support the use of unregistered `Transports`.

15.1 Identifiers for Transports

`Transports` **must** be registered with an `Identifier` in the `ORI Naming Environment`. The structure of these `Identifiers` **must** be as specified in the remainder of this Section.

Table 15.1 shows the components used to construct `ORI Identifiers` for `Transports`. These components represent:

- The `Naming Environment` of the `Identifier`, which **must** be 'ori'
- The Namespace dedicated to `Identifiers` of `Transports` in the `ORI Naming Environment`, which **must** be 'tsp'
- A name assigned on registration to indicate the network protocol that is used by the `Transport`.
- A name assigned on registration to indicate the actual `Transport`.

Table 15.1: Construction of ORI Identifiers for Transports

Naming Environment	Name-space	Network Protocol	Transport
ori	**tsp**	**http**	**openurl-by-val**
ori	**tsp**	**https**	**openurl-by-ref**
ori	**tsp**	**http**	**openurl-01**

The ORI `Identifier` for a `Transport` is obtained by concatenating the various components, separated by colons. For example:

- **ori:tsp:http:openurl-by-val** is the `Identifier` of the registered `Transport` that uses HTTP as the network protocol and the `By-Value OpenURL` actually to transport `ContextObjects` over HTTP
- **ori:tsp:https:openurl-by-ref** is the `Identifier` of the registered `Transport` that uses HTTPS as the network protocol and the `By-Reference OpenURL` actually to transport `ContextObjects` over HTTPS

- **ori:tsp:http:openurl-01** is the `Identifier` of a registered `Transport` that uses HTTP as the network protocol and the OpenURL 0.1 specification as the actual `Transport`.

`Identifiers` of `Transports` will not be used in actual representations of the `ContextObject`. They are introduced to support `Registry` management and unambiguous expression of `Community Profiles`.

16. Community Profiles [Registry]

When implementing this Standard, representatives of specific application domains make choices from Registry entries, so defining a Community Profile. A Community Profile specifies a consistent selection from among the registered properties of the OpenURL Framework to support a specific application domain:

- A single ContextObject Format. Because of the nature of ContextObject Formats, this choice brings along a selection of:
  - A set of constraints on the type and amount of Entities, and Descriptors used for the representation of a ContextObject
  - A constraint on the amount of ContextObjects that can be represented in a single instance document that conforms to the ContextObject Format
  - A single Physical Representation
  - A single Constraint Language
  - One or more Character Encodings
- Metadata Formats that may be used for By-Value Metadata or By-Reference Metadata descriptions or both. Because of the nature of Metadata Formats, this choice brings along a selection of:
  - One or more Physical Representations
  - One or more Constraint Languages
  - One or more Character Encodings
- URI Namespaces that may be used to describe Entities by an Identifier Descriptor
- ORI Namespaces that may be used to describe Entities by an Identifier Descriptor
- One or more Transports that specify how representations of ContextObjects in the chosen ContextObject Format must be transported

A Community creating a Community Profile **may** register a required property if it is not available in the Registry.

Part 2, Section 12, defines the Community Profiles that form the initial content of the Registry. Through the registration mechanism, Community Profiles **may** be added to the Registry. This Standard does not support the use of unregistered Community Profiles.

16.1 Identifiers for Community Profiles

Community Profiles **must** be registered with an Identifier in the ORI Naming Environment. The structure of these Identifiers **must** be as specified in the remainder of this Section.

Table 16.1 shows the components used to construct ORI Identifiers for Community Profiles. These components represent:

- The Naming Environment of the Identifier, which **must** be ‘ori’

- The Namespace dedicated to `Identifiers` of `Community Profiles` in the `ORI Naming Environment`, which **must** be 'pro'
- A name assigned on registration to indicate the `Community Profile`

Table 16.1: Construction of ORI Identifiers for Community Profiles

Naming Environment	Name-space	Name
ori	**pro**	**sap0**
ori	**pro**	**sap1**
ori	**pro**	**sap2**

The ORI `Identifier` for a `Community Profile` is obtained by concatenating the various components, separated by colons. For example:

- **ori:pro:sap0** is the `Identifier` of the registered "San Antonio Level 0" `Community Profile` that is introduced to facilitate ongoing support of the OpenURL 0.1 specification
- **ori:pro:sap1** is the `Identifier` of the registered "San Antonio Level 1" `Community Profile` that builds on the KEV `ContextObject Format` and includes choices for `Metadata Formats` and ORI Namespaces of `Identifiers` focused on scholarly publications
- **ori:pro:sap2** is the `Identifier` of the registered "San Antonio Level 2" `Community Profile` that builds on the XML `ContextObject Format` and includes further choices for `Metadata Formats` and ORI Namespaces of `Identifiers` focused on scholarly publications

`Identifiers` of `Community Profiles` will not be used in actual representations of the `ContextObject`. They are introduced to support `Registry` management and unambiguous identification of `Community Profiles`.

16.2. Expression of Community Profiles

`Community Profiles` **must** be expressed by means of an XML instance document that is valid according to an XML Schema of that will be defined in the near future. In addition to this **mandatory** 'machine readable' expression of a `Community Profile`, it is strongly **recommended** that Communities create a 'human readable' description to support a better understanding of the `Community Profile` by implementers.