ANSI/NISO Z39.99-2014 **ISSN: 1041-5653**

ResourceSync Framework Specification

Abstract: This ResourceSync specification describes a synchronization framework for the web consisting of various capabilities that allow third-party systems to remain synchronized with a server's evolving resources. The capabilities may be combined in a modular manner to meet local or community requirements. This specification also describes how a server should advertise the synchronization capabilities it supports and how third-party systems may discover this information. The specification repurposes the document formats defined by the Sitemap protocol and introduces extensions for them.

An American National Standard
Developed by the
National Information Standards Organization

Approved by the
American National Standards Institute (ANSI)
April 21, 2014

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

About NISO Standards

NISO standards are developed by Working Groups 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.

These standards may be revised or withdrawn at any time.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

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

ISSN: 1041-5653 (National Information standards series)
ISBN: 978-1-937522-20-9 (HTML)
ISBN: 978-1-937522-19-3 (PDF)

Contents

Foreword

(This foreword is not part of the *ResourceSync Framework Specification*, ANSI/NISO Z39.99-2014. It is included for information only.)

About This Standard

This document is structured as follows:

- Section 1, *Introduction,* discusses the purpose and scope and provides example use cases and includes a *Walkthrough*, introducing the core components of the ResourceSync framework by example.
- Section 2, *Normative References*, lists references to other standards/specifications that are required for conformance with this standard.
- Section 3, *Definitions*, introduces terminology important for the understanding of this standard.
- Section 4, *Namespace Prefix Bindings*, identifies the namespaces used for this specification.
- Section 5, *Synchronization Processes*, provides a high-level overview of the various ResourceSync capabilities and shows how these fit in processes aimed at remaining in step with resource changes.
- Section 6, *Framework Organization*, describes the overall organization of the ResourceSync framework: structure, navigation, and discovery.
- Section 7, *Sitemap Document Formats*, describes the way in which the document formats introduced by the Sitemap protocol are used to convey synchronization related information in all ResourceSync capabilities.
- Section 8, *Describing the Source*, and Section 9, *Advertising Capabilities*, describe how a server should convey the ResourceSync capabilities it supports.
- Section 10, *Describing Resources*, Section 11, *Packaging Resources*, Section 12, *Describing Changes*, Section 13, Packaging Changes, and Section 14, *Linking to Related Resources*, each provide details about a capability that a server may implement in order to enable remote systems to remain aligned with its evolving data.
- Appendix A provides an overview of the requirements for use of the `at` and `from` attributes in ResourceSync documents.

Trademarks, Services Marks

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

NISO Voting Members

At the time this standard was balloted and approved, the following were members of the NISO Z39.99-201x ResourceSync voting pool:

American Institute of Physics (AIP)
Evan Owens (primary), Aravind Akella (alternate)

American Library Association (ALA)
Nancy Kraft

American Library Association (ALA)
Nancy Kraft

American Theological Library Association (ATLA)
Brenda Bailey-Hainer (primary),
Maria Stanton (alternate)

Association for Information Science & Technology (ASIS&T)
Mark Needleman

Association of Research Libraries (ARL)
Julia Blixrud

EBSCO Information Services
Oliver Pesch

EnvisionWare, Inc.
Robert Walsh (primary), Michael Monk (alternate)

Ex Libris, Inc.
Mike Dicus

Innovative Interfaces, Inc.
Brad Jung (primary), John McCullough (alternate)

International DOI Foundation (IDF)
Norman Paskin

ITHAKA/JSTOR/Portico
Amy Kirchhoff (primary), Bruce Heterick (alternate)

John Wiley & Sons, Ltd.
Duncan Campbell (primary),
Craig Van Dyck (alternate)

Library of Congress
Sally McCallum (primary), John Zagas (alternate)

Los Alamos National Laboratory
Dianna Magnoni (primary),
Herbert Van de Sompel (alternate)

Microsoft Corporation
Alex Wade

Minitex
Cecelia Boone (primary), Valerie Horton (alternate), Paul Swanson (alternate)

Music Library Association
Nara Newcomer (primary),
David Sommerfield (alternate)

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

National Federation of Advanced Information Services (NFAIS)
Marjorie Hlava

National Library of Finland
Juha Hakala

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

OCLC Online Computer Library Center
Tam Dalrymple (primary),
Thomas Hickey (alternate)

SAGE Publications
Carol Richman

Scholarly iQ
Gary Van Overborg (primary),
John Milligan (alternate)

Statewide California Electronic Library Consortium (SCELC)
Rick Burke

The Library Corporation (TLC)
Juli Marsh (primary), Wayne Hicks (alternate)

NISO D2D Topic Committee

At the time NISO approved this standard, the following individuals served on the Discovery to Delivery (D2D) Topic Committee that had oversight for this project.

Kristin Antelman
North Carolina State University Libraries

Pascal Calarco, Co-chair
University of Waterloo Library

Lucy Harrison, Co-chair
Florida Virtual Campus

Juli Marsh
The Library Corporation (TLC)

Peter Murray
Lyrasis

Tim Shearer
University of North Carolina Chapel Hill Libraries

Christine Stohn
Ex Libris, Inc.

Chris Shillum
Reed Elsevier

NISO ResourceSync Working Group Members

The following individuals served on the NISO ResourceSync Working Group, which developed and approved this standard:

Todd Carpenter, Co-chair
NISO

Bernhard Haslhofer
Vienna University of Technology

Richard Jones
Cottage Labs

Martin Klein
Los Alamos National Laboratory

Graham Klyne
University of Oxford

Carl Lagoze
University of Michigan

Stuart Lewis
JISC Collections

Peter Murray
Lyrasis

Michael Nelson
Old Dominion University

Shlomo Sanders
Ex Libris, Inc.

Robert Sanderson
Los Alamos National Laboratory

Herbert Van de Sompel, Co-chair
Los Alamos National Laboratory

Paul Walk
EDINA

Simeon Warner
Cornell University

Zhiwu Xie
Virginia Tech University Libraries

Jeff Young
OCLC Online Computer Library Center

Acknowledgements

This specification is the collaborative work of NISO and the Open Archives Initiative. Funding for ResourceSync is provided by the Alfred P. Sloan Foundation. UK participation is supported by Jisc.

We also thank numerous individual contributors including: Martin Haye (California Digital Library), David Rosenthal (Stanford University Libraries), Ed Summers (Library of Congress), and Vincent Wehren (Microsoft).

ResourceSync Framework Specification

1 Introduction

1.1 Purpose and Scope

The web is highly dynamic, with resources continuously being created, updated, and deleted. As a result, using resources from a remote server involves the challenge of remaining in step with its changing content. In many cases, there is no need to reflect a server's evolving content perfectly and, therefore, well-established resource discovery techniques, such as crawling, suffice as an updating mechanism. However, there are significant use cases that require low latency and high accuracy in reflecting a remote server's changing content. These requirements have typically been addressed by ad-hoc technical approaches implemented within a small group of collaborating systems. There have been no widely adopted, web-based approaches.

This ResourceSync specification introduces a range of easy to implement capabilities that a server may support in order to enable remote systems to remain more tightly in step with its evolving resources. It also describes how a server should advertise the capabilities it supports. Remote systems may inspect this information to determine how best to remain aligned with the evolving data.

Each capability provides a different synchronization functionality, such as a list of the server's resources or its recently changed resources, including what the nature of the change was: create, update, or delete. All capabilities are implemented on the basis of the document formats introduced by the Sitemap protocol. Capabilities may be combined to achieve varying levels of functionality and hence meet different local or community requirements. This modularity provides flexibility and makes ResourceSync suitable for a broad range of use cases.

1.2 Motivating Examples

Many projects and services have synchronization needs and have implemented ad hoc solutions. ResourceSync provides a standard synchronization method that will reduce implementation effort and facilitate easier reuse of resources. This section describes motivating examples with differing needs and complexities.

Consider first the case of a website for a small museum collection. The website may contain just a few dozen static webpages. The maintainer may create a Resource List (see Section 10.1) of these webpages and expose it to services that leverage ResourceSync.

When building services over Linked Data, it is often desirable to maintain a local copy of data for improved access and availability. Harvesting may be enabled by publishing a Resource List for the dataset. In many cases, resource representations exposed as Linked Data are small and so retrieving them via individual HTTP GET requests is slow because of the large number of round trips for a small amount of content. Publishing a Resource Dump (see Section 11.1) that points to content packaged and described in ZIP files makes this more efficient for the client and less burdensome for the server. Continued synchronization is enabled by recurrently publishing an up-to-date Resource List or Resource Dump, or, more efficiently, by publishing a Change List (see Section 12.1) that provides information about resource changes only.

The arXiv.org collection of scientific articles propagates resource changes to a set of mirror sites and interacting services on a daily basis. As of July 2013, the collection contains about 2.6 million resources and there are about 1,600 changes (creates, updates) per day. The mirroring system operated since 1994 uses HTTP with custom change descriptions and occasionally rsync to verify the copies and to cope with any errors in the incremental updates. The approach assumes a tight

connection between arXiv.org and its mirrors. It would be desirable to have a solution that allows any third-party system to accurately synchronize with arXiv.org using commodity software. arXiv.org could publish both metadata records and full-text content as separate web resources with their own URI. Use of ResourceSync capabilities including Resource Lists (see Section 10.1), Resource Dumps (see Section 11.1), Change Lists (see Section 12.1), and Change Dumps (see Section 13.1), allows both mirrors and new parties to remain accurately in sync with the collection. This would extend the openly available metadata sharing capabilities provided by arXiv.org, currently implemented via OAI-PMH, to full-text sharing in a web-friendly fashion.

1.3 Walkthrough

Let's assume a Source, http://example.com/, that exposes changing content that others would like to remain synchronized with. A first step towards making this easy for Destinations is for the Source to publish a Resource List (see Section 10.1) that conveys the URIs of resources available for synchronization. This Resource List is expressed as a Sitemap. As shown in Example 1, the Source conveys the URI of each resource as the value of the `<loc>` child element of a `<url>` element. Note the `<rs:md>` child element of the `<urlset>` root element, which expresses that the Sitemap implements ResourceSync's Resource List capability. It also conveys that the Resource List reflects the state of the Source's resources at the datetime provided in the `at` attribute. This datetime allows a Destination to quickly determine whether it has previously processed this specific Resource List.

```
<?xml version="1.0" encoding="UTF-8"?>
<urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9"
        xmlns:rs="http://www.openarchives.org/rs/terms/">
  <rs:md capability="resourcelist"
         at="2013-01-03T09:00:00Z"/>
  <url>
      <loc>http://example.com/res1</loc>
  </url>
  <url>
      <loc>http://example.com/res2</loc>
  </url>
</urlset>
```

Example 1: A Resource List

The Source can provide additional information in the Resource List to help the Destination optimize the process of collecting content and verifying its accuracy. For example, when the Source expresses the datetime of the most recent modification for a resource, a Destination can determine whether or not it already holds the current version, minimizing the number of HTTP requests it needs to issue in order to remain up-to-date. Example 2 shows this information conveyed using Sitemap's `<lastmod>` element. When the Source also conveys a hash for a specific bitstream, a Destination can verify whether the process of obtaining it was successful. The example shows this information conveyed using the `hash` attribute on the `<rs:md>` element. In addition, the Source can provide links to related resources using the `<rs:ln>` element. The example shows a link to a mirror copy of the second listed resource, indicating that the Source would prefer a Destination to obtain the resource from it.

```
<?xml version="1.0" encoding="UTF-8"?>
<urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9"
        xmlns:rs="http://www.openarchives.org/rs/terms/">
  <rs:md capability="resourcelist"
         at="2013-01-03T09:00:00Z"/>
  <url>
      <loc>http://example.com/res1</loc>
      <lastmod>2013-01-02T13:00:00Z</lastmod>
      <rs:md hash="md5:1584abdf8ebdc9802ac0c6a7402c03b6"/>
  </url>
  <url>
      <loc>http://example.com/res2</loc>
      <lastmod>2013-01-02T14:00:00Z</lastmod>
      <rs:md hash="md5:1e0d5cb8ef6ba40c99b14c0237be735e"/>
      <rs:ln rel="duplicate"
             href="http://mirror.example.com/res2"/>
  </url>
</urlset>
```

Example 2: A Resource List with additional information

In order to describe its changing content in a more timely manner, the Source can increase the frequency at which it publishes an up-to-date Resource List. However, changes may be so frequent or the size of the content collection so vast that regularly updating a complete Resource List may be impractical. In such cases, the Source can implement an additional capability that communicates information about changes only. To this end, ResourceSync introduces Change Lists. A Change List enumerates resource changes, along with the nature of the change (create, update, or delete) and the time that the change occurred. A Destination can recurrently obtain a Change List from the Source, inspect the listed changes to discover those it has already acted upon, and process the remaining ones. Changes in a Change List are provided in forward chronological order, making it straightforward for a Destination to determine which changes it already processed. In addition, a Change List also contains datetimes that convey the start time and the end time of the temporal interval covered by the Change List. These times convey that all resource changes that occurred during the interval are described in the Change List. (ResourceSync does not specify for how long change lists must continue to be available once they have been produced. The longer that Change Lists are maintained by the Source, the better the odds are for a Destination to catch up on changes it missed because it was offline, for example.)

Example 3 shows a Change List. The value of the `capability` attribute of the `<rs:md>` child element of `<urlset>` makes it clear that, this time, the Sitemap is a Change List and not a Resource List. The `from` and `until` attributes inform about the temporal interval covered by the Change List. The Change List shown below conveys two resource changes, one an update and the other a deletion, as can be seen from the value of the `change` attribute of the `<rs:md>` element. The example also shows the use of the `<lastmod>` element to convey the time of the changes. Note that these times are used to order the Change List chronologically.

```
<?xml version="1.0" encoding="UTF-8"?>
<urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9"
        xmlns:rs="http://www.openarchives.org/rs/terms/">
  <rs:md capability="changelist"
         from="2013-01-02T00:00:00Z"
         until="2013-01-03T00:00:00Z"/>
  <url>
      <loc>http://example.com/res2.pdf</loc>
      <lastmod>2013-01-02T13:00:00Z</lastmod>
      <rs:md change="updated"/>
  </url>
  <url>
      <loc>http://example.com/res3.tiff</loc>
      <lastmod>2013-01-02T18:00:00Z</lastmod>
      <rs:md change="deleted"/>
  </url>
</urlset>
```

Example 3: A Change List

A Destination can issue HTTP GET requests against each resource URI listed in a Resource List. For large Resource Lists, issuing all of these requests may be cumbersome. Therefore, ResourceSync introduces a capability that a Source can use to make packaged content available. A Resource Dump, implemented as a Sitemap, contains pointers to packaged content. Each content package referenced in a Resource Dump is a ZIP file that contains the Source's bitstreams along with a Resource Dump Manifest that describes each. The Resource Dump Manifest itself is also implemented as a Sitemap. A Destination can retrieve a Resource Dump, obtain content packages by dereferencing the contained pointers, and unpack the retrieved packages. Since the Resource Dump Manifest also lists the URI the Source associates with each bitstream, a Destination is able to achieve the same result as obtaining the data by dereferencing the URIs listed in a Resource List. Example 4 shows a Resource Dump that points at a single content package. Dereferencing the URI of that package leads to a ZIP file that contains the Resource Dump Manifest shown in Example 5. It indicates that the Source's ZIP file contains two bitstreams. The `path` attribute of the `<rs:md>` element conveys the file path of the bitstream in the ZIP file (the relative file system path where the bitstream would reside if the ZIP were unpacked), whereas the `<loc>` element conveys the URI associated with the bitstream at the Source.

An additional capability, the Change Dump, provides a functionality similar to a Resource Dump but pertains to packaging bitstreams of resources that have changed during a temporal interval, instead of packaging a snapshot of resource bitstreams at a specific moment in time.

```
<?xml version="1.0" encoding="UTF-8"?>
<urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9"
        xmlns:rs="http://www.openarchives.org/rs/terms/">
  <rs:md capability="resourcedump"
         at="2013-01-03T09:00:00Z"/>
  <url>
      <loc>http://example.com/resourcedump.zip</loc>
      <lastmod>2013-01-03T09:00:00Z</lastmod>
  </url>
</urlset>
```

Example 4: A Resource Dump

```
<?xml version="1.0" encoding="UTF-8"?>
<urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9"
        xmlns:rs="http://www.openarchives.org/rs/terms/">
  <rs:md capability="resourcedump-manifest"
         at="2013-01-03T09:00:00Z"/>
  <url>
      <loc>http://example.com/res1</loc>
      <lastmod>2013-01-03T03:00:00Z</lastmod>
      <rs:md hash="md5:1584abdf8ebdc9802ac0c6a7402c03b6"
             path="/resources/res1"/>
  </url>
  <url>
      <loc>http://example.com/res2</loc>
      <lastmod>2013-01-03T04:00:00Z</lastmod>
      <rs:md hash="md5:1e0d5cb8ef6ba40c99b14c0237be735e"
             path="/resources/res2"/>
  </url>
</urlset>
```

Example 5: A Resource Dump Manifest detailing the content of a ZIP file

ResourceSync also introduces a Capability List, which is a way for the Source to describe the capabilities it supports for one set of resources. Example 6 shows an example of such a description. It indicates that the Source supports the Resource List, Resource Dump, and Change List capabilities and it lists their respective URIs. Note the inclusion of an `<rs:ln>` child element of `<urlset>` that links by means of a `describedby` relation to a description of the set of resources covered by the Capability List. Because these capabilities are conveyed in the same Capability List, they uniformly apply to this set of resources. For example, if a given resource appears in the Resource List then it must also appear in a Resource Dump and changes to the resource must be reported in the Change List.

```
<?xml version="1.0" encoding="UTF-8"?>
<urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9"
        xmlns:rs="http://www.openarchives.org/rs/terms/">
  <rs:ln rel="describedby"
         href="http://example.com/info_about_set1_of_resources.xml"/>
  <rs:ln rel="up"
         href="http://example.com/resourcesync_description.xml"/>
  <rs:md capability="capabilitylist"/>
  <url>
      <loc>http://example.com/dataset1/resourcelist.xml</loc>
      <rs:md capability="resourcelist"/>
  </url>
  <url>
      <loc>http://example.com/dataset1/resourcedump.xml</loc>
      <rs:md capability="resourcedump"/>
  </url>
  <url>
      <loc>http://example.com/dataset1/changelist.xml</loc>
      <rs:md capability="changelist"/>
  </url>
</urlset>
```

Example 6: A Capability List enumerating the ResourceSync capabilities a Source supports for a set of its resources

There are three ways by which a Destination can discover whether and how a Source supports ResourceSync: a Source-wide approach, a resource-specific approach, and an approach that leverages existing practice for discovering Sitemaps. The Source-wide approach leverages the well-known URI specification and consists of the Source making a Source Description, like the one shown

in Example 7, available at `/.well-known/resourcesync`. The Source Description enumerates the Capability Lists a Source offers, one Capability List per set of resources. If a Source only has one set of resources and hence only one Capability List, the mandatory Source Description contains only one pointer. The resource-specific discovery approach consists of a Source providing a link in an HTML document or in an HTTP Link header that points at a Capability List that covers the resource that provides the link. Note in Example 6, the inclusion of an `<rs:ln>` child element of `<urlset>` that links by means of an `up` relation to the Source Description, allowing for navigation from a Capability List to a Source Description. Yet another approach follows the established practice for discovering Sitemaps via a Source's `robots.txt` file. Since a Resource List is a Sitemap it can be made discoverable by including its URI in the `robots.txt` file as the value of the `Sitemap` directive. A navigational `up` link included in the Resource List allows discovery of a Capability List pertaining to the set of resources covered by that Resource List, and a further `up` link in the Capability List leads to the Source Description.

```
<?xml version="1.0" encoding="UTF-8"?>
<urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9"
        xmlns:rs="http://www.openarchives.org/rs/terms/">
  <rs:ln rel="describedby"
         href="http://example.com/info-about-source.xml"/>
  <rs:md capability="description"/>
  <url>
      <loc>http://example.com/dataset1/capabilitylist.xml</loc>
      <rs:md capability="capabilitylist"/>
      <rs:ln rel="describedby"
             href="http://example.com/info_about_set1_of_resources.xml"/>
  </url>
</urlset>
```

Example 7: A Source Description with a pointer to the Capability List for the single set of resources offered by a Source

In some cases, there is a need to split the documents described so far into parts. For example, the Sitemap protocol currently prescribes a maximum of 50,000 resources per Sitemap and a Source may have more resources that are subject to synchronization. The ResourceSync framework follows these community defined limits and hence, in such cases, publishes multiple Resource Lists as well as a Resource List Index that points to each of them. The Resource List Index is expressed using Sitemap's `<sitemapindex>` document format. Example 8 shows a Resource List Index that points at two Resource Lists.

```
<?xml version="1.0" encoding="UTF-8"?>
<sitemapindex xmlns="http://www.sitemaps.org/schemas/sitemap/0.9"
              xmlns:rs="http://www.openarchives.org/rs/terms/">
  <rs:md capability="resourcelist"
         at="2013-01-03T09:00:00Z"/>
  <sitemap>
      <loc>http://example.com/resourcelist-part1.xml</loc>
  </sitemap>
  <sitemap>
      <loc>http://example.com/resourcelist-part2.xml</loc>
  </sitemap>
</sitemapindex>
```

Example 8: A Resource List Index expressed using the <sitemapindex> document format

2 Normative References

The following documents contain provisions that are required for implementing this standard. All standards are subject to revision; the most current version of these standards should be used.

[ALE] Snell, J. *Atom Link Extensions*. Internet Draft. Internet Engineering Task Force (IETF), June 8, 2012. Available at: http://tools.ietf.org/html/draft-snell-atompub-link-extensions-09

[IANA MIME] MIME Media Types [registry website]. Internet Assigned Numbers Authority (IANA). Available at: http://www.iana.org/assignments/media-types

[IANA Relation] Link Relations [registry website]. Internet Assigned Numbers Authority (IANA). Available at: http://www.iana.org/assignments/link-relations/link-relations.xml

[Memento] Van de Sompel, H., M. L. Nelson, and R. D. Sanderson. *HTTP framework for time-based access to resource states – Memento*. Internet Draft. Internet Engineering Task Force (IETF), October 1, 2013. Available at: http://tools.ietf.org/html/draft-vandesompel-memento-10

[RFC 2616] Fielding, R. et al. *Hypertext Transfer Protocol -- HTTP/1.1*. RFC 2616. Internet Engineering Task Force (IETF), June 1999. Available at: http://www.ietf.org/rfc/rfc2616.txt

[RFC 4287] Nottingham, M., and R. Sayre, eds. The Atom Syndication Format. RFC 4287. Internet Engineering Task Force (IETF), December 2005. Available at: http://www.ietf.org/rfc/rfc4287.txt

[RFC 5988] Nottingham, M. *Web Linking*. RFC 5988. Internet Engineering Task Force (IETF), October 2010. Available at: http://www.ietf.org/rfc/rfc5988.txt

[RFC 6249] Bryan, A. et al. *Metalink/HTTP: Mirrors and Hashes*. RFC 6249. Internet Engineering Task Force (IETF), June 2011. Available at: http://www.ietf.org/rfc/rfc6249.txt

[RFC 6906] Wilde, E. The 'profile' Link Relation Type. RFC 6906. Internet Engineering Task Force (IETF), March 2013. Available at: http://www.ietf.org/rfc/rfc6906.txt

[Sitemaps] *Sitemaps XML Format*. sitemaps.org, last updated February 27, 2008. Available at: http://www.sitemaps.org/protocol.html

[W3C Datetime] Wolf, Misha, and Charles Wicksteed. *Date and Time Formats*. W3C Note. World Wide Web Consortium, August 27, 1998. Available at: http://www.w3.org/TR/1998/NOTE-datetime-19980827

[ZIP] *.ZIP File Format Specification*. Application Note. Version 6.3.3. PKWARE Inc., September 1, 2012. Available at: http://www.pkware.com/documents/casestudies/APPNOTE.TXT

3 Definitions

The following terms, as used in this standard, have the meanings indicated.

Term	Definition
Source	A server that hosts resources subject to synchronization.
Destination	A system that synchronizes itself with the Source's resources.
set of resources	A collection of resources that is made available for synchronization by a Source. A Source may expose one or more such collections and support distinct ResourceSync capabilities for each. Individual resources may be included in more than one set of resources.

This specification uses the terms resource, representation, request, response, content negotiation, client, and server as described in Architecture of the World Wide Web.

4 Namespace Prefix Bindings

Throughout this document, the following namespace prefix bindings are used:

Prefix	Namespace URI	Description
(*none*)	`http://www.sitemaps.org/schemas/sitemap/0.9`	Sitemap XML elements defined in the Sitemap protocol
`rs`	`http://www.openarchives.org/rs/terms/`	Namespace for elements introduced in this specification

5 Synchronization Processes

Section 1.3 provides a concrete walkthrough of some capabilities that a Source may implement and describes how a Destination may use those capabilities to remain synchronized with the Source's changing data. This section provides a high-level overview of the various ResourceSync capabilities and shows how these fit into processes at a Destination designed to keep it in step with changes.

5.1 Source Perspective

From the perspective of a **Source**, the ResourceSync capabilities that may be supported to enable Destination processes to remain in sync with its changing data are summarized as follows:

- **Describing Content** – In order to describe its data, a Source may maintain an up-to-date Resource List. A basic Resource List minimally provides the URIs of resources that the Source makes available for synchronization. However, additional information may be added to the Resource List to optimize the Destination's process of obtaining the Source's resources, including the most recent modification time of resources and fixity information such as content-based checksum or hash and length. Figure 1 shows a Source publishing up-to-date Resource Lists at times t2 and t4. At t4, too many resources need to be listed to fit in a single Resource List and hence multiple Resource Lists are published and grouped in a Resource List Index.
- **Packaging Content** – In order to make its data available for download, a Source may recurrently make an up-to-date Resource Dump of its content available. A Resource Dump points at one or more packages, each of which contains bitstreams associated with resources hosted by the Source. Each package also contains a Resource Dump Manifest that provides metadata about the bitstreams contained in the package, and must minimally include their associated URI and their file path in the ZIP file. Figure 1 shows a Source publishing up-to-date Resource Dumps at times t1 and t3. At time t3, multiple Resource Dumps are published and grouped in a Resource Dump Index.
- **Describing Changes** – In order to achieve lower synchronization latency and/or to improve transfer efficiency, a Source may publish a Change List that provides information about changes to its resources. It is up to the Source to decide the temporal interval covered by a Change List, for example, all the changes that occurred during the previous hour, the current day, or since the most recent publication of a Resource List. For each resource change, a Change List must minimally convey the URI of the changed resource as well as the datetime

and nature of the change (create, update, delete). Since a Change List is organized on the basis of changes, it may list the same resource multiple times, once per change. Figure 2 shows three Change Lists. The first Change List covers resource changes that occurred between t1 and t3, the second between t3 and t5, and the third between t5 and t7. Since too many changes occurred between t5 and t7 to fit in a single Change List, multiple Change Lists are published and grouped in a Change List Index.

- **Packaging Changes** – In order to make content changes available for download, a Source may publish a Change Dump. A Change Dump points at one or more packages, each of which contains bitstreams that correspond to the state of resources after they changed. Each package also contains a Change Dump Manifest that provides metadata about the bitstreams provided in the Change Dump. For each bitstream, the Change Dump Manifest must minimally include the associated URI, the datetime when the change that resulted in the bitstream occurred, the nature of the change (create, update, delete) and, where appropriate, the file path of the bitstream in the ZIP file. It is up to a Source to decide the temporal interval covered by a Change Dump, for example, covering all the resource changes that occurred during the previous hour, the current day, or since the most recent publication of a Resource Dump. Since a Change Dump is organized on the basis of changes, the package(s) it points at may contain multiple bitstreams associated with any given resource, one per change. Figure 2 shows three Change Dumps. The first Change Dump covers resource changes that occurred between t2 and t4, the second between t4 and t6, and the third between t6 and t8. During the time period between t6 and t8, multiple Change Dumps are published and grouped in a Change Dump Index.
- **Linking to Related Resources** – There are several reasons to provide additional links from a resource subject to synchronization to related resources, including:
    - **Alternate Content Transfer** – The default mechanisms by which a Destination obtains content for a resource are to issue an HTTP GET against its URI found in a Resource List or Change List, or to unpack packages obtained via a Resource Dump or Change Dump. However, additional approaches may also be supported. For example, a Source may prefer, for synchronization purposes, that content be obtained from a mirror server and hence from a different URI. Also, a Source may allow obtaining only the changes that a resource underwent, instead of the entire changed resource. This may be desirable when the resource size is considerable and/or the frequency of changes high. Such an Alternate Content Transfer approach is expressed by means of a link from the resource to another resource that makes the content available in an alternate way. It is possible that certain Destinations do not recognize a specific Alternate Content Transfer approach, in which case ignoring the link and dereferencing the resource's URI remains the fallback approach.
    - **Resources and Metadata about Resources** – Cases exist where both resources and metadata about resources must be synchronized, for example, a collection of scientific publications and metadata describing each. From the ResourceSync perspective, both the resource and the metadata about it are regarded as resources with distinct URIs that are subject to synchronization. Their inter-relationship is expressed by means of links with appropriate relation types.
    - **Prior Versions of Resources** – In some cases a Destination requires a copy of each version of a resource, not just the most recent one. A Source may support discovery and access to prior resource versions through links. Three approaches are provided, one based on linking to resource versions, and two that leverage features of the Memento protocol for time-based access to resource states.

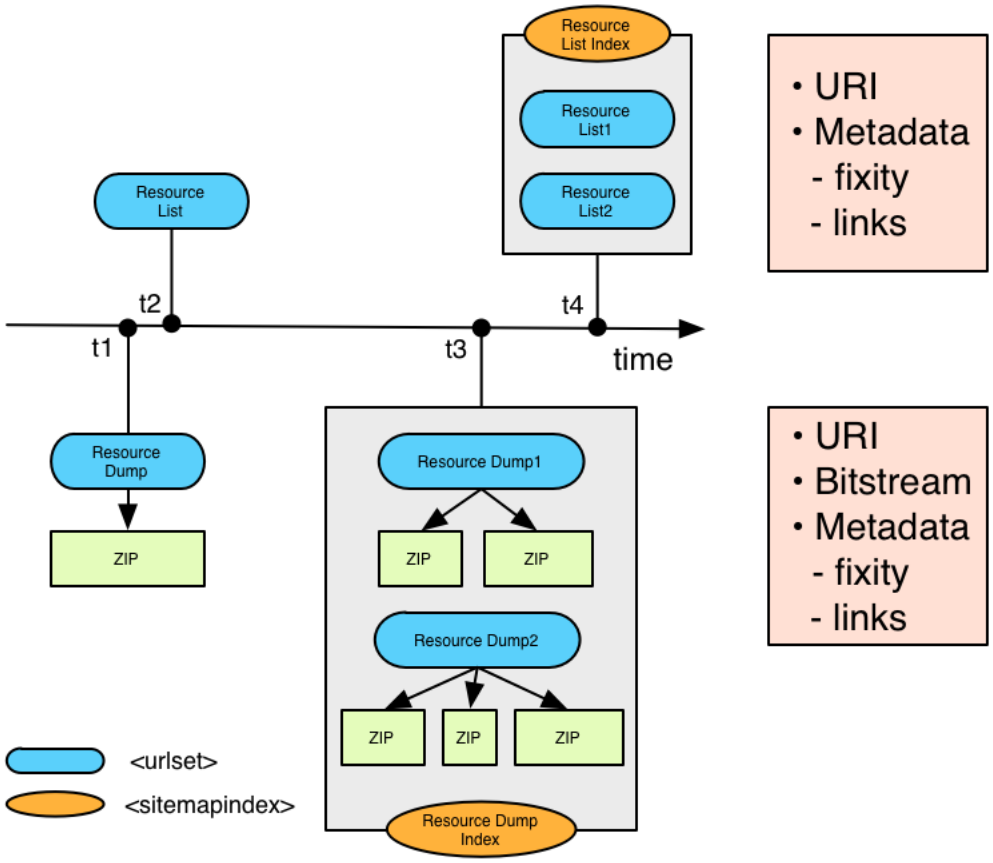

Figure 1: ResourceSync Source perspective of resource description

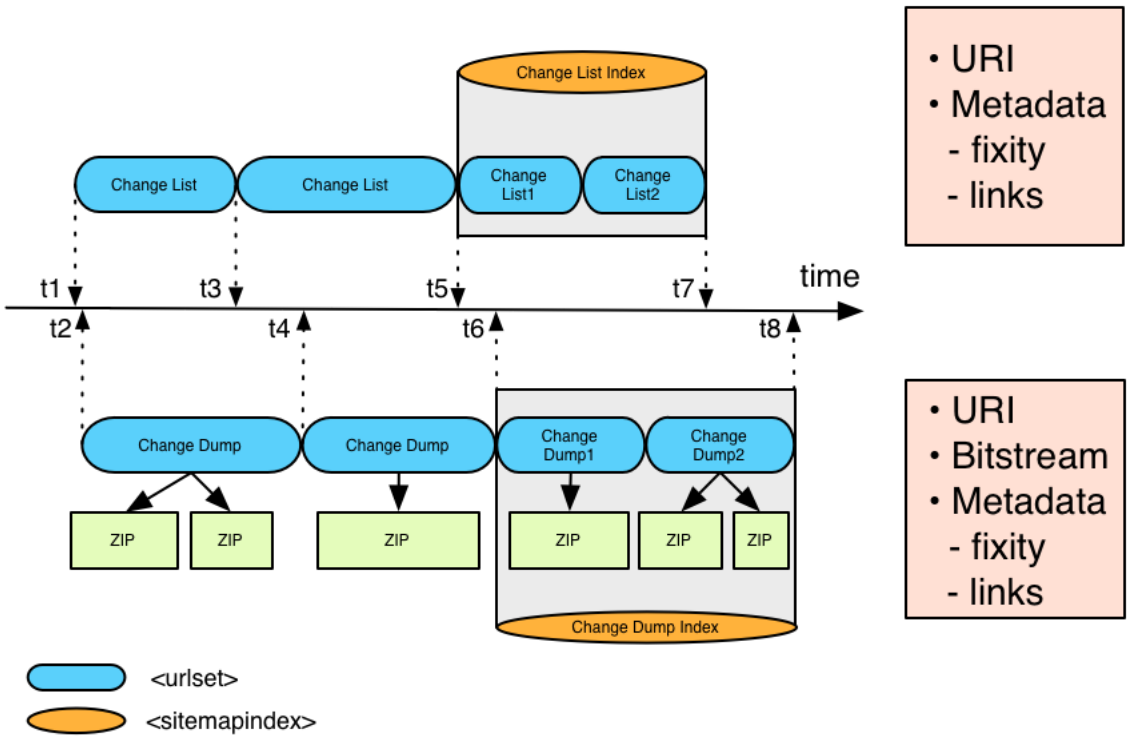

Figure 2: ResourceSync Source perspective of change description

5.2 Destination Perspective

From the perspective of a **Destination**, three key processes are enabled by the ResourceSync capabilities; Figure 3 provides an overview:

- **Baseline Synchronization** – In order to become synchronized with a Source, the Destination must make an initial copy of the Source's data. A Destination may obtain the Resource List that conveys the URIs of the Source's resources, and subsequently dereference those URIs one by one. A Destination may also obtain a Resource Dump that conveys the URIs of one or more content packages each of which contains bitstreams associated with the Source's resources. A Destination may dereference those URIs and subsequently unpack the retrieved content packages, guided by the contained Resource Dump Manifest.
- **Incremental Synchronization** – A Destination may remain in sync with a Source by repeatedly performing a Baseline Synchronization. To increase efficiency and decrease latency, a Source may communicate information about changes to its resources via Change Lists. This allows a Destination to obtain up-to-date content by dereferencing the URIs of newly created and updated resources listed in the Change List. It also allows a Destination to remove its copies of deleted resources, if needed. A Source may also make a Change Dump available that points at one or more packages, each of which contains bitstreams that correspond to the state of resources after they changed. In this case the Destination first obtains the Change Dump, then obtains the package(s) by dereferencing the URI(s) listed in the Change Dump, and subsequently unpacks those, guided by the contained Change Dump Manifest.
- **Audit** – In order to verify whether it is in sync with the Source, a Destination must be able to check that the content it obtained matches the current resources hosted by the Source both regarding coverage and accuracy. This requires an up-to-date list of resources hosted by the

Source, which may be compiled on the basis of a Resource List and Change Lists. It also requires these Lists to contain metadata per resource that characterizes its most recent state, such as last modification time, length, and content-based hash.

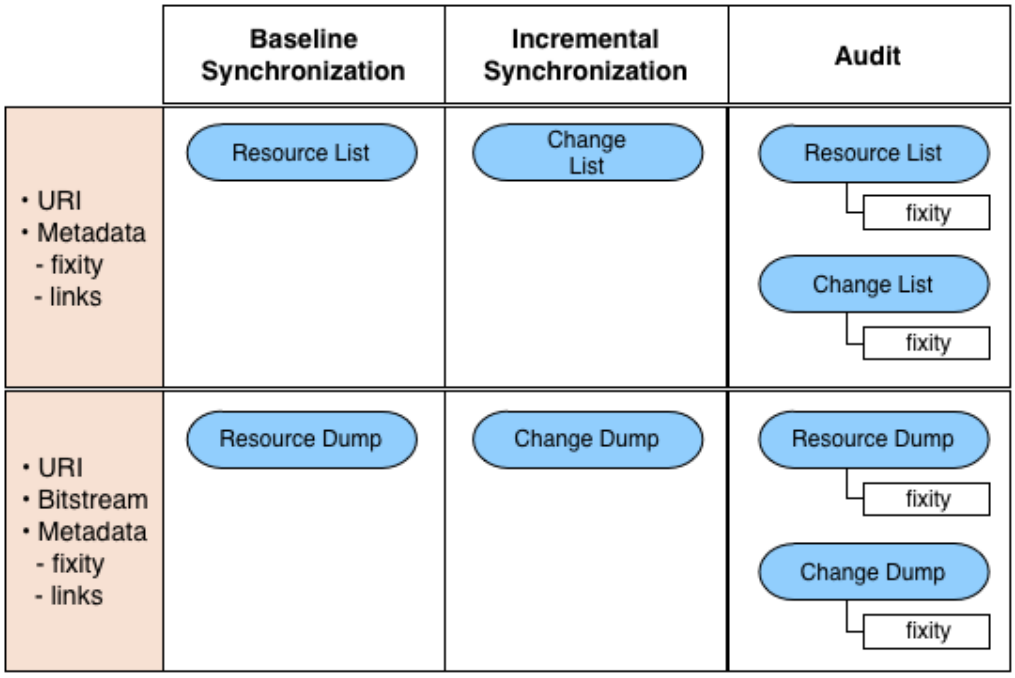

Figure 3: ResourceSync Destination perspective

5.3 Summary

Table 1 provides a summary of Section 5. The table lists Destination processes as columns and Source capabilities as rows, with cells indicating the applicability of a capability for a given process.

Table 1: Source capabilities versus Destination processes

Source Capabilities	Destination Processes		
	Baseline Synchronization	Incremental Synchronization	Audit
Describing the Source (see Section 8)	X	X	X
Advertising Capabilities (see Section 9)	X	X	X
Describing Resources (see Section 10)			
Resource List (see Section 10.1)	X		X
Packaging Resources (see Section 11)			
Resource Dump (see Section 11.1)	X		
Describing Changes (see Section 12)			

Source Capabilities	Destination Processes		
	Baseline Synchronization	Incremental Synchronization	Audit
Change List (see Section 12.1)		X	X
Packaging Changes (see Section 13)			
Change Dump (see Section 13.1)		X	
Linking to Related Resources (see Section 14)			
Mirrored Content (see Section 14.2)	X	X	X
Alternate Representations (see Section 14.3)	X	X	X
Patching Content (see Section 14.4)		X	X
Resources and Metadata about Resources (see Section 14.5)	X	X	X
Prior Versions of Resources (see Section 14.6)		X	X
Collection Membership (see Section 14.7)	X	X	X
Republishing Resources (see Section 14.8)	X	X	X

6 Framework Organization

6.1 Structure

All capabilities in the ResourceSync framework are implemented on the basis of the `<urlset>` and `<sitemapindex>` Sitemap document formats. Figure 4: ResourceSync framework structure, depicts the overall structure of the set of documents that is used:

- At the top of the picture is the mandatory **Source Description**. It is a Destination's typical entry point to learn about a Source's ResourceSync implementation. The Source Description enumerates all Capability Lists offered by the Source, one Capability List per set of resources. If the Source only offers one set of resources, the ResourceSync Description contains a single pointer. A Source Description is expressed as a `<urlset>` document and, per Capability List, a `<url>` element is introduced. The `<loc>` child element of `<url>` contains the URI of the Capability List, and the `capabilitylist` value for the `capability` attribute of the `<rs:md>` child element of `<url>` makes clear that the URI is that of a Capability List.
- A **Capability List** enumerates all capabilities supported for a set of the Source's resources. The capabilities defined in this ResourceSync specification are Resource List, Change List, Resource Dump, and Change Dump. Additional capabilities may be defined in other specifications. A Capability List is expressed as a `<urlset>` document and, for each supported capability, a `<url>` element is introduced. The `<loc>` child element of `<url>` contains the URI of the document that implements a capability, and the type of capability is expressed by means of the value of the `capability` attribute of the `<rs:md>` child element of `<url>`, e.g., `resourcelist` for a Resource List.

- A **Resource List** and a **Change List** point at resources. A representation of a resource may be obtained by dereferencing its URI, listed as the value of the `<loc>` child element of the `<url>` element for the resource.
- A **Resource Dump** and a **Change Dump** point at packages, each of which contains bitstreams associated with resources, as well as a Manifest that describes the bitstreams provided in the package.
- The **Manifest** contained in a package of bitstreams is expressed as a `<urlset>` document. For each bitstream contained in the package, that document contains a `<url>` element; the `<loc>` child element of `<url>` provides the URI that corresponds to the bitstream, whereas the `path` attribute of the `<rs:md>` child element of `<url>` provides the path of the bitstream in the package.
- If a single document suffices to express a Source Description, a Resource List, a Change List etc., then the `<urlset>` document format is used. If multiple documents are required, each is expressed using the `<urlset>` document format, and a `<sitemapindex>` document is introduced as an index to point at all individual `<urlset>` documents. As a result, the URI of, for example, a Resource List provided in a Capability List may be either that of a `<urlset>` or a `<sitemapindex>` document. The `<urlset>` or `<sitemapindex>` documents used for a specific capability (e.g., Resource List) have the same value for the `capability` attribute (e.g., `resourcelist`).

The Resource List branch of Figure 4 is fully compatible with the existing Sitemap specification, whereas the other branches are extensions introduced to support resource synchronization that leverage the Sitemap document formats.

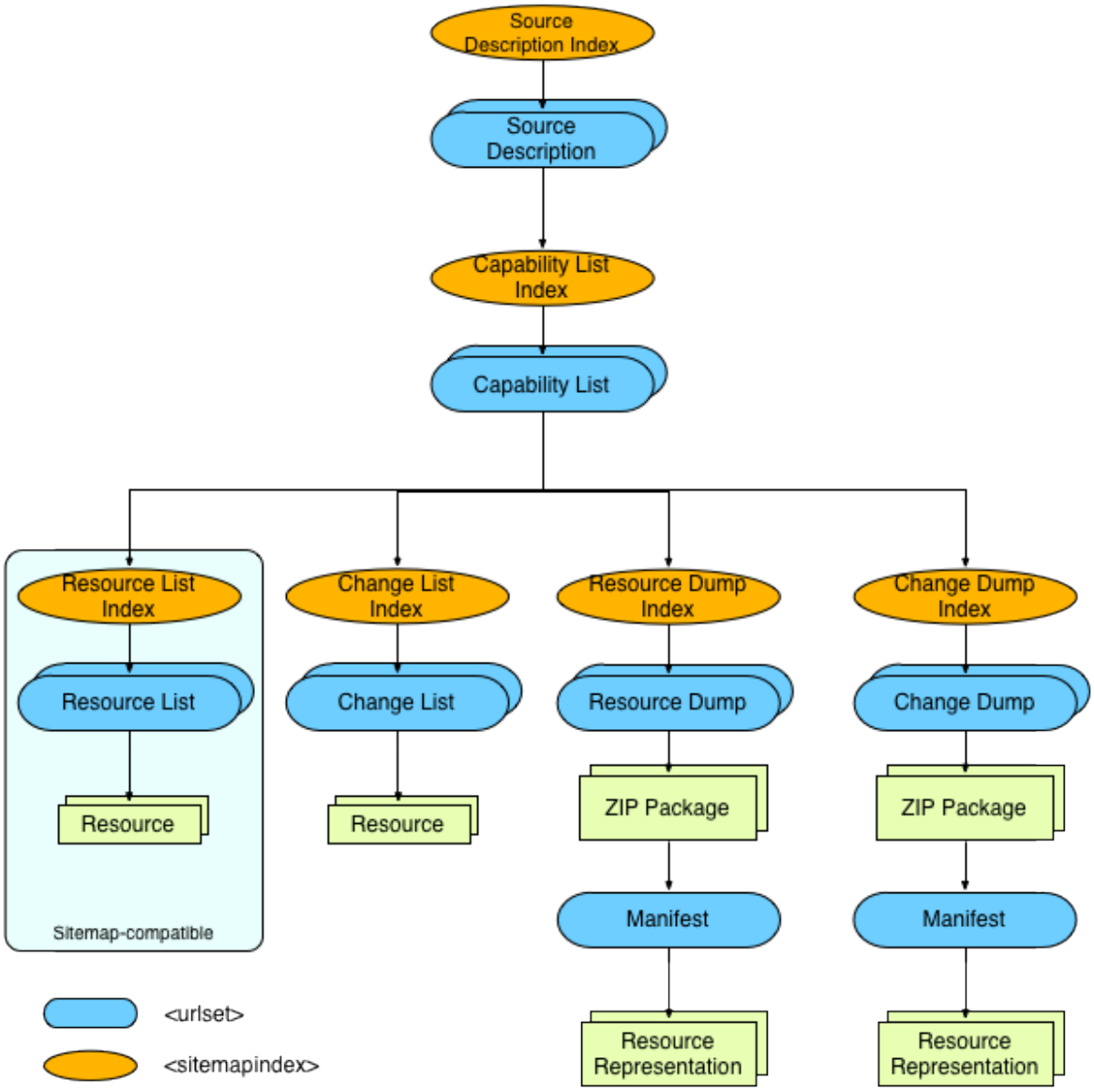

Figure 4: ResourceSync framework structure

6.2 Navigation

The following mechanisms are introduced to support navigating the document hierarchy described in Section 6.1; they are illustrated in Figure 5 and Figure 6:

- A link for upward navigation is provided by means of an `<rs:ln>` child element of the `<urlset>` or `<sitemapindex>` element of a document. This pointer has `up` as the value for the `rel` attribute, and the URI of the document that sits higher in the hierarchy is provided as the value of the `href` attribute. Following consecutive `up` links eventually leads to the Source Description.
- A link for navigation from a document to the index under which it resides, if one exists, is provided by means of an `<rs:ln>` child element of the `<urlset>` element of the document.

This pointer has `index` as the value for the `rel` attribute, and the URI of the index document is provided as the value of the `href` attribute.

- A link for downward navigation is provided by the content of a `<url>` element: the URI of the document that sits lower in the hierarchy is provided as the value of its `<loc>` child element, and its type is conveyed as the value of the `capability` attribute of the `<rs:md>` child element.

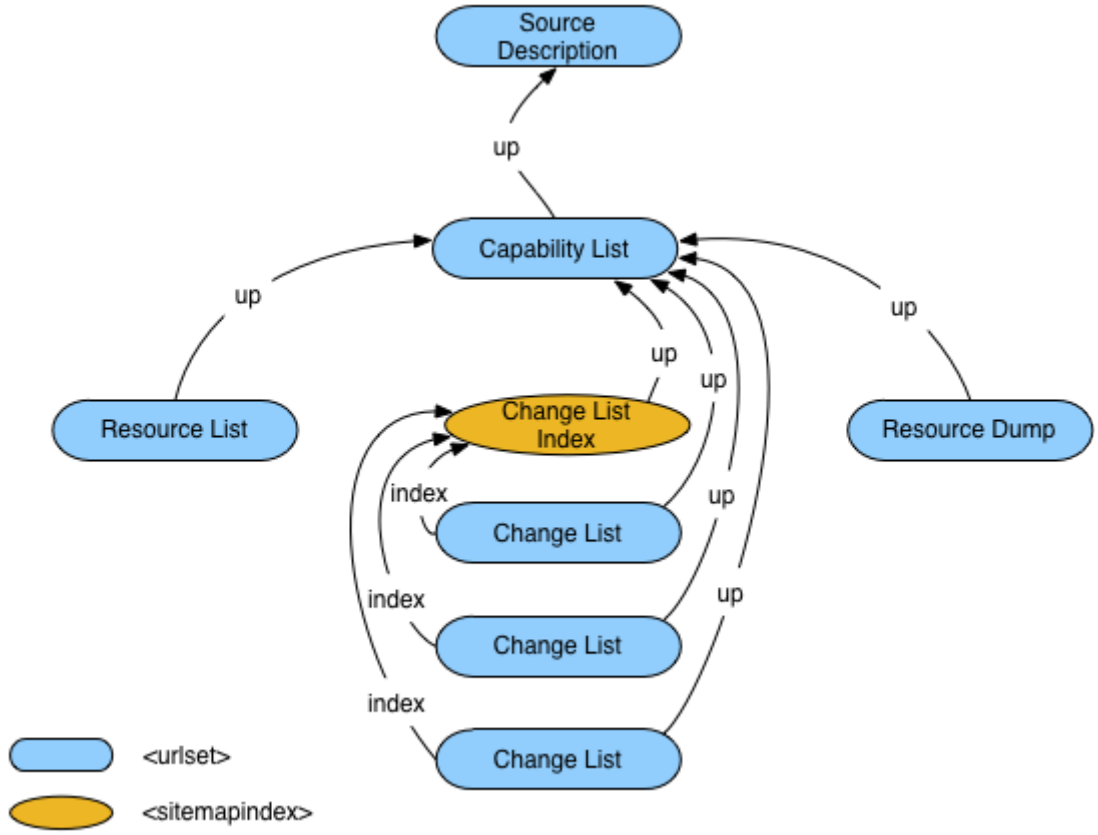

Figure 5: ResourceSync upwards navigation

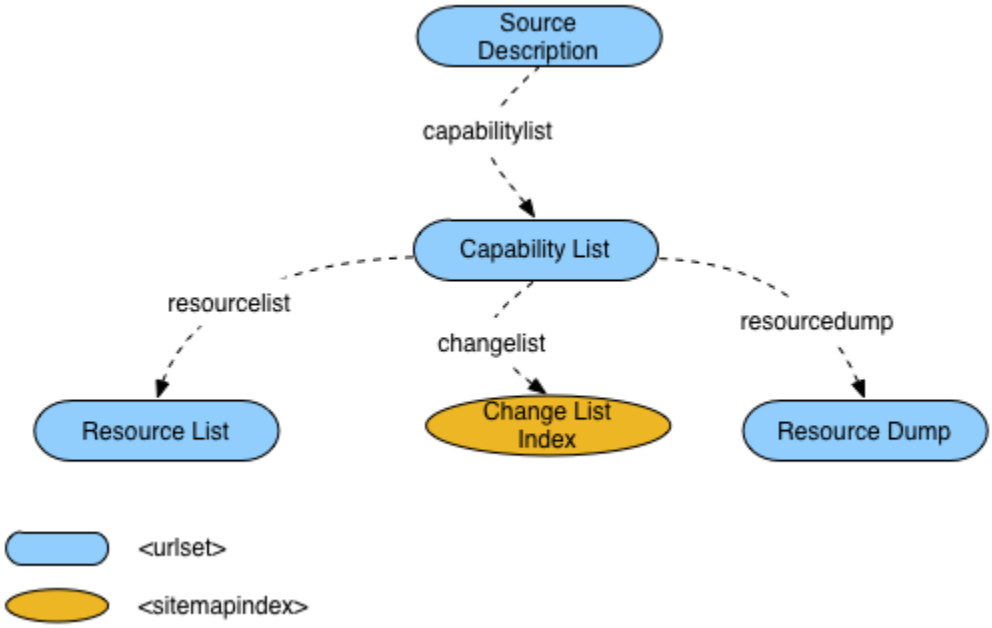

Figure 6: ResourceSync downwards navigation

6.3 Discovery

6.3.1 Overview

ResourceSync provides three ways for a Destination to discover whether and how a Source supports ResourceSync: a Source-wide approach detailed in Section 6.3.2, a resource-specific approach detailed in Section 6.3.3, and an approach that leverages the existing practice of Sitemap discovery via the `robots.txt` file described in Section 6.3.4. All approaches are summarized in Figure 7.

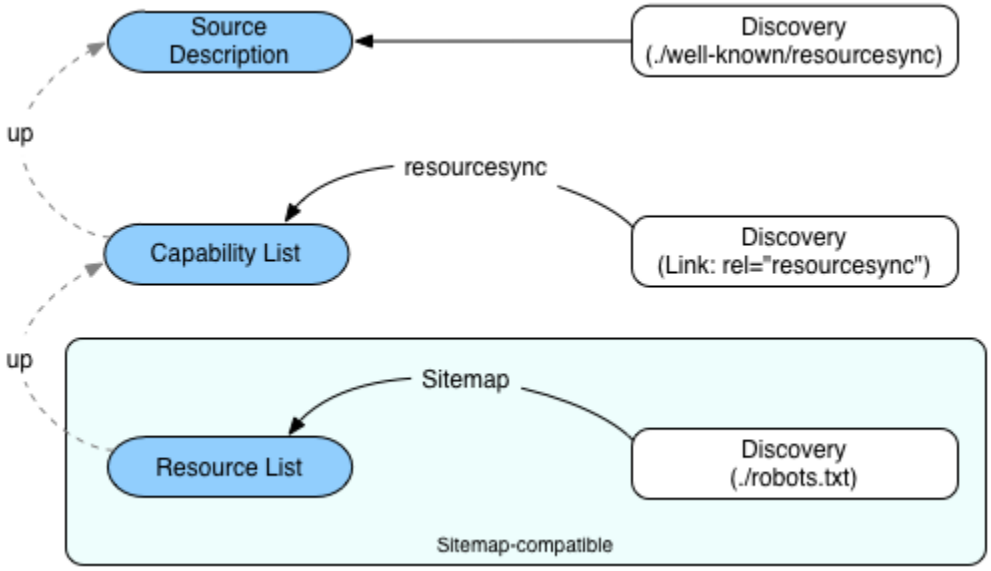

Figure 7: Discovery of Source Description and Capability List

6.3.2 ResourceSync Well-Known URI

A Source must publish a Source Description, such as the one shown in Example 7, and it should be published at the well-known URI [RFC 5785] `/.well-known/resourcesync` as defined here. The Source Description document enumerates a Source's Capability Lists and as such is an appropriate entry point for Destinations interested in understanding a Source's capabilities.

6.3.3 Links

A Capability List may be made discoverable by means of links provided either in an HTML document [HTML Links, XHTML Links] or in an HTTP Link header [RFC 5988].

In order to include a discovery link in an HTML document, a `<link>` element is introduced in the `<head>` of the document that points to a Capability List. This `<link>` must have a `rel` attribute with a value of `resourcesync`. The Capability List that is made discoverable in this way must pertain to the resource that provides the link. This means that the resource must be covered by the capabilities listed in the linked Capability List. Example 9 shows the structure of a webpage that contains a link to a Capability List.

As shown in Example 6 the Source Description can be discovered from the Capability List by following the link provided in the `<rs:ln>` element with the relation type `up`.

```
<html>
  <head>
    <link rel="resourcesync"
          href="http://www.example.com/dataset1/capabilitylist.xml"/>
    ...
  </head>
  <body>...</body>
</html>
```

Example 9: Discovery by means of an HTML link

A Capability List may also be made discoverable by means of an HTTP Link header that may be included with a representation of a resource of any content-type. In order to do so, a link is introduced in the HTTP Link header. The target of this link is the URI of a Capability List and the value of its `rel` attribute is `resourcesync`. The Capability List that is made discoverable in this way must pertain to the resource that provides the link. This means that the resource must be covered by the capabilities listed in the linked Capability List. Example 10 shows an excerpt of an HTTP response header that illustrates this approach.

As shown in Example 6 the Source Description can be discovered from the Capability List by following the link provided in the `<rs:ln>` element with the relation type `up`.

```
HTTP/1.1 200 OK
  Date: Thu, 21 Jan 2010 00:02:12 GMT
  Server: Apache
  Link: <http://www.example.com/dataset1/capabilitylist.xml>;
        rel="resourcesync"
  ...
```

Example 10: Discovery by means of an HTTP link

6.3.4 robots.txt

A Resource List is a Sitemap and hence may be made discoverable via the established approach of adding a `Sitemap` directive to a Source's `robots.txt` file that has the URI of the Resource List as its value. If a Source supports multiple sets of resources, multiple directives may be added, one for each Resource List associated with a specific set of resources. In case a Source supports both

regular Sitemaps and ResourceSync Sitemaps (Resource Lists) they may be made discoverable, again, by including multiple `Sitemap` directives as shown in Example 11.

Once a Resource List for a set of resources has been discovered in this manner, the corresponding Capability List can be discovered by following a link with the `up` relation type provided in the Resource List. Next, the Source Description can be discovered by following yet another link with the `up` relation type provided in the Capability List.

```
User-agent: *
Disallow: /cgi-bin/
Disallow: /tmp/
Sitemap: http://example.com/dataset1/resourcelist.xml
```

Example 11: A robots.txt file that points at a Resource List

7 Sitemap Document Formats

In order to convey information pertaining to resources in the ResourceSync framework, the Sitemap (root element `<urlset>`) and Sitemap index (root element `<sitemapindex>`) document formats introduced by the Sitemap protocol are used for a variety of purposes. The `<sitemapindex>` document format is used when it is necessary to group multiple documents of the `<urlset>` format. The ResourceSync framework follows community-defined limits for when to publish multiple documents of the `<urlset>` format. At time of publication of this specification, the limit is 50,000 items per document and a document size of 50 MB.

The document formats, as well as their ResourceSync extension elements, are shown in Table 2. The `<rs:md>` and `<rs:ln>` elements are introduced to express metadata and links, respectively. Both are in the ResourceSync XML Namespace and may have attributes. The attributes of these elements defined by ResourceSync are listed in Table 3 and detailed below. As shown in the examples, these attributes must not have an XML Namespace prefix. The `<rs:ln>` element as well as several of the ResourceSync attributes are based upon other specifications and in those cases inherit the semantics defined there; the “Specification” column of Table 3 refers to those specifications. Communities may introduce additional attributes when needed but must use an XML Namespace other than that of ResourceSync and must appropriately use namespace prefixes for those attributes.

Table 2: The Sitemap document formats including the ResourceSync extensions

Sitemap	Sitemap Index

`<?xml version="1.0" encoding="UTF-8"?>`
`<urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9"`
`xmlns:rs="http://www.openarchives.org/rs/terms/">`
`<rs:md />`
`<rs:ln />`
`<url>`
`<loc />`
`<lastmod />`
`<rs:md />`
`<rs:ln />`
`</url>`
`<url>`
`...`
`</url>`
`</urlset>`
	`<?xml version="1.0"`
	`encoding="UTF-8"?>`
	`<sitemapindex xmlns="..."`
	`xmlns:rs="...">`
	`<rs:md />`
	`<rs:ln />`
	`<sitemap>`
	`<loc />`
	`<lastmod />`
	`<rs:md />`
	`<rs:ln />`
	`</sitemap>`
	`<sitemap>`
	`...`
	`</sitemap>`
	`</sitemapindex>`

The overall structure of the ResourceSync documents is as follows:

<urlset> or <sitemapindex> – These elements are the root elements of ResourceSync documents; this specification adds one mandatory and one optional child element to the child elements of the Sitemap document formats:

- <rs:md> – In this context, the element conveys information about the document itself. Its use is mandatory and it may have the following attributes:
  - at – This attribute is used for Resource Lists, Resource List Indexes, Resource Dumps, Resource Dump Indexes, and Resource Dump Manifests; it is not used for Change Lists, Change List Indexes, Change Dumps, Change Dump Indexes, and Change Dump Manifests. Required use of the attribute is detailed in the sections describing the respective documents and summarized in Appendix A. The at attribute conveys the datetime at which the process of taking a snapshot of resources for their inclusion in the document to which the attribute pertains started. It thus provides a guarantee that each resource state represented in the document is the result of all changes to the resource at least up until the datetime expressed as the value of the at attribute. The attribute value is expressed as a W3C Datetime and the use of a complete date and time expressed in UTC using the format YYYY-MM-DDThh:mm:ss[.s]Z is recommended. The attribute represents the time of a snapshot such as t1, t2, t3, and t4 of Figure 1.
  - capability – This attribute is mandatory in all ResourceSync documents. The value of the attribute conveys the nature of the document, e.g., whether the document is a Resource List, a Change List, a Manifest, etc. Defined values are resourcelist, changelist, resourcedump, changedump, resourcedump-manifest, changedump-manifest, capabilitylist, and description.
  - completed – This optional attribute is used for Resource Lists, Resource List Indexes, Resource Dumps, Resource Dump Indexes, and Resource Dump Manifests; it is not used for Change Lists, Change List Indexes, Change Dumps, Change Dump Indexes, and Change Dump Manifests. The completed attribute conveys the datetime at which the process of taking a snapshot of resources for their inclusion in the document to which the attribute pertains completed. The combination of the datetimes provided in the at and completed attributes expresses an interval during which resources may have changed beyond the state they had at the datetime expressed in the at attribute. The attribute value for completed is expressed as a W3C Datetime and the use of a complete date and time expressed in UTC using the format YYYY-MM-DDThh:mm:ss[.s]Z is recommended.
  - from – This attribute is used for Change Lists, Change List Indexes, Change Dumps, Change Dump Indexes, and Change Dump Manifests; it is not used for Resource Lists, Resource List Indexes, Resource Dumps, Resource Dump Indexes, and Resource Dump Manifests. Required use of the attribute is detailed in the sections describing the respective documents and summarized in Appendix A. The attribute indicates that all changes that occurred to the set of resources at the Source since the datetime expressed (and up until the datetime expressed in the until attribute, if it exists) are included in the document to which the attribute pertains. The attribute value is expressed as a W3C Datetime and the use of a complete date and time expressed in UTC using the format YYYY-MM-DDThh:mm:ss[.s]Z is recommended. For example, the first Change List in Figure 2 would have a from value of t1, and the second Change List would have a from value of t3.
  - until – This optional attribute is used for Change Lists, Change List Indexes, Change Dumps, Change Dump Indexes, and Change Dump Manifests; it is not used for Resource Lists, Resource List Indexes, Resource Dumps, Resource Dump Indexes, and Resource Dump Manifests. The attribute indicates that all changes that occurred to the

set of resource at the Source up until the datetime expressed are included in the document to which the attribute pertains.

- When a document carries the `until` attribute, this indicates that the document will not be updated anymore.
- When a change document does not carry the `until` attribute, any subsequent changes to the corresponding set of resources will cause the document to be updated.

The attribute value is expressed as a W3C Datetime and the use of a complete date and time expressed in UTC using the format `YYYY-MM-DDThh:mm:ss[.s]Z` is recommended. For example, the first Change List in Figure 2 would have an `until` value of t3 and the second Change List would have an `until` value of t5.

- `<rs:ln>` – A repeatable element used to support discovery of other documents by means of a link. Required use of the element is detailed in the sections that describe the documents in which `<rs:ln>` is used. It may have several attributes and the ones defined by ResourceSync are as follows:
  - `href` – A mandatory attribute to convey the URI of the other document.
  - `rel` – A mandatory attribute to express a relationship. The following values are explicitly used in this specification:
    - `describedby` – for linking from a Capability List to a document that describes the set of resources covered by it, and from a Source Description to a document that describes the Source.
    - `up` – for linking from a Capability List to the Source Description and from a document that conveys a capability, such as a Resource List, to the Capability List under which it resides.
    - `index` – for linking from a document that conveys a capability (e.g., a Resource List) to a parent index document (e.g., a Resource List Index).
- `<url>` or `<sitemap>` – The `<urlset>` element may have zero or more `<url>` child elements, and the `<sitemapindex>` element has zero or more `<sitemap>` child elements. Each such child element is used to convey information about a resource that plays a role in the ResourceSync framework. They may have the following child elements:
  - `<loc>` – A mandatory element that conveys the URI of the resource that plays a role in the ResourceSync framework.
  - `<lastmod>` – An element that conveys the last modification time of the resource with the URI provided in `<loc>`, expressed as a W3C Datetime as described earlier in this section. Its use is mandatory in Change Lists and Change Dump Manifests, and optional in other documents.
  - `<changefreq>` – An optional element that provides a hint about the change frequency of the resource with the URI provided in `<loc>`. Defined values are `always`, `hourly`, `daily`, `weekly`, `monthly`, `yearly`, and `never`. The value `always` should be used for resources that change each time they are accessed. The value `never` should be used for archived resources.
  - `<rs:md>` – In this context, the element conveys metadata pertaining to the resource with the URI provided in `<loc>`. The element is not repeatable, and is mandatory for some documents and optional for others, as described in the appropriate sections. It may have several attributes and the ones defined by ResourceSync are as follows:

- `at` and `completed` – The semantics and value of these attributes are as defined earlier in this section. They are only used for Resource List Indexes, Resource Dumps, and Resource Dump Indexes. Required use of the attributes is detailed in the sections describing the respective documents and summarized in Appendix A.
- `capability` – This attribute is mandatory in Source Descriptions and Capability Lists. Its value indicates the nature of the resource identified by the URI in the `<loc>` element, e.g., a Resource List, a Change List, a Change Dump, etc. Values defined in this specification are `resourcelist`, `changelist`, `resourcedump`, `changedump`, and `capabilitylist`.
- `change` – The value of the attribute conveys the type of change that a resource underwent. Values defined in this specification are `created`, `updated`, and `deleted` to convey the creation, update, and deletion of a resource, respectively. This attribute is used in Change Lists (Section 12.1) and Change Dump Manifests (Section 13.2).
- `encoding` – The value of the attribute conveys what content codings have been applied to the resource. The value of the `encoding` attribute should be equal to the value of the `content-encoding` header in the HTTP response as defined in RFC 2616, Sec. 14.11.
- `from` and `until` – The semantics and value of these attributes are as defined earlier in this section. They are only used for Change List Indexes, Change Dumps, and Change Dump Indexes. Required use of the attributes is detailed in the sections describing the respective documents and summarized in Appendix A.
- `hash` – The value of the attribute conveys fixity information for a resource representation returned when the URI in `<loc>` is dereferenced. The attribute value is expressed in the form of a whitespace-delimited list of hash values. Each hash value is represented by a hex-encoded digest and is preceded by a token that identifies the utilized hash algorithm, e.g., `md5:, sha-256:`.
- `length` – The value of the attribute conveys the content length of a resource representation returned when the URI in `<loc>` is dereferenced. The value of the `length` attribute should be equal to the value of the `Content-Length` header in the HTTP response and must be computed as defined in RFC 2616, Sec. 4.4.
- `path` – The attribute is only used in Resource Dump Manifests (Section 11.2) and Change Dump Manifests (Section 13.2). Its value conveys the file path of the bitstream associated with the URI in `<loc>` in the ZIP file. That is the relative file system path where the bitstream would reside if the ZIP were unpacked.
- `type` – The value of the attribute conveys the Media Type of a resource representation returned when the URI in `<loc>` is dereferenced. Registered values are listed in the IANA MIME Media Type registry.

o `<rs:ln>` – In this context, an optional and repeatable element used to link to resources related to the one with the URI provided in `<loc>`, such as a copy on a mirror site, a prior version of the resource, etc. (see *Linking to Related Resources* in Section 5.1). It may have several attributes and the ones defined by ResourceSync are as follows:

- `href` – A mandatory attribute to convey the URI of the related resource.
- `rel` – A mandatory attribute to convey the relationship between the resource with the URI in `<loc>` and the one with the URI in `href`. Values for the `rel` attribute are listed in the document *Relation Types Used in the ResourceSync Framework*. The ones featured in this specification are:

  - `contents` – for a link from an entry in a Resource Dump or Change Dump that points to a bitstream package to a Resource Dump Manifest or a Change Dump Manifest, respectively, for that bitstream package.
  - `duplicate` – for a link to a resource's mirror location.
  - `alternate` and `canonical` – for a link to an alternate representation of a resource.
  - `http://www.openarchives.org/rs/terms/patch` – for a link to a resource that details the difference between the previous and current version of a resource.
  - `describedby` and `describes` – for a link providing additional information about a resource.
  - `memento` and `timegate` – for a link to access prior versions of a resource.
  - `collection` – for a link that expresses collection membership.
  - `via` – for a link that provides provenance information.
- `encoding`, `hash`, `length`, `modified`, `path`, `type` – Optional attributes with meanings as described earlier in this section and pertaining to the related resource.
- `pri` – An optional attribute used to express a priority among links with the same relation type. The attribute value is an integer between 1 and 999,999, with a lower integer indicating a higher priority and the absence of the attribute indicating a value of 999,999.

Table 3 lists the elements used in ResourceSync documents and for each shows the attributes defined by ResourceSync that may be used with them. The "Specification" column refers to the specification where elements or attributes were introduced that ResourceSync equivalents are based upon and inherit their semantics from. A mark in the "Representation" column for an attribute indicates that it should only be used when a specific representation of a resource is concerned, whereas a mark in the "Resource" column indicates it is usable for a resource in general. A W3C XML Schema (available at http://www.niso.org/schemas/resourcesync) is provided to validate the elements introduced by ResourceSync.

Relation types other than the ones listed above may be used in the ResourceSync Framework. Valid relation types must be registered in the *IANA Link Relation Type Registry* or expressed as URIs as specified in RFC 5988, Sec. 4.2. The document *Relation Types Used in the ResourceSync Framework* attempts to provide an up-to-date overview.

Table 3: Elements and associated attributes defined for the ResourceSync documents

Element/Attribute	Specification	Resource	Representation
`<urlset>` or `<sitemapindex>`	Sitemap protocol		
`<rs:md>`	This specification		
`at`	This specification		
`capability`	This specification		
`completed`	This specification		
`from`	This specification		
`until`	This specification		
`<rs:ln>`	RFC4287		
`href`	RFC4287		
`rel`	RFC4287		

Element/Attribute	Specification	Resource	Representation
<url> or <sitemap>	Sitemap protocol		
<loc>	Sitemap protocol		
<lastmod>	Sitemap protocol		
<changefreq>	Sitemap protocol		
<rs:md>	This specification		
at	This specification		
capability	This specification		
change	This specification	X	X
completed	This specification		
encoding	RFC2616		X
from	This specification		
hash	Atom Link Extensions		X
length	RFC4287		X
path	This specification		X
type	RFC4287		X
until	This specification		
<rs:ln>	This specification		
encoding	RFC2616		X
hash	Atom Link Extensions		X
href	RFC4287	X	X
length	RFC4287		X
modified	Atom Link Extensions	X	X
path	This specification		X
pri	RFC6249	X	X
rel	RFC4287	X	X
type	RFC4287		X

8 Describing the Source

A **Source Description** is a mandatory document that enumerates the Capability Lists offered by a Source. Since a Source has one Capability List per set of resources that it distinguishes, the Source Description will enumerate as many Capability Lists as the Source has distinct sets of resources.

The Source Description is based on the <urlset> format. It has the <urlset> root element and the following structure:

- The mandatory <rs:md> child element of <urlset> must have a capability attribute that has a value of description.
- A recommended <rs:ln> child element of <urlset> with the relation type describedby points to a document that provides information about the Source.
- One <url> child element of <urlset> should be included for each Capability List offered by the Source. This element does not have attributes, but uses child elements to convey information about the Capability Lists. The <url> element has the following child elements:
  - A mandatory <loc> child element provides the URI of the respective Capability List.

- An optional <rs:md> child element must have a `capability` attribute with a value of `capabilitylist` to convey that the URI points to a Capability List.
- A recommended <rs:ln> child element with a `describedby` relation type points to a document that describes the set of resources described by the Capability List.

The <lastmod> elements should be omitted from the Source Description unless the Source updates the Source Description every time it updates one of the Capability Lists.

Example 12 shows a Source Description where the Source offers three Capability Lists.

```
<?xml version="1.0" encoding="UTF-8"?>
<urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9"
        xmlns:rs="http://www.openarchives.org/rs/terms/">
  <rs:ln rel="describedby"
         href="http://example.com/info_about_source.xml"/>
  <rs:md capability="description"/>
  <url>
      <loc>http://example.com/capabilitylist1.xml</loc>
      <rs:md capability="capabilitylist"/>
      <rs:ln rel="describedby"
             href="http://example.com/info_about_set1_of_resources.xml"/>
  </url>
  <url>
      <loc>http://example.com/capabilitylist2.xml</loc>
      <rs:md capability="capabilitylist"/>
      <rs:ln rel="describedby"
             href="http://example.com/info_about_set2_of_resources.xml"/>
  </url>
  <url>
      <loc>http://example.com/capabilitylist3.xml</loc>
      <rs:md capability="capabilitylist"/>
      <rs:ln rel="describedby"
             href="http://example.com/info_about_set3_of_resources.xml"/>
  </url>
</urlset>
```

Example 12: A Source Description

If a Source needs to or chooses to publish multiple Source Descriptions, it must group them by means of a Source Description Index.

9 Advertising Capabilities

A **Capability List** is a document that enumerates all capabilities supported by a Source for a specific set of resources. The Source defines which resources are part of the set of resources described by the Capability List. If there is more than one such set, then the Source must distinguish them with different capability lists. The choice of which resources are part of which set may derive from a variety of criteria, including media type, collection membership, change frequency, subject of the resource and many others.

A Capability List points at the capability documents for its set of resources: Resource List (Section 10.1), Resource Dump (Section 11.1), Change List (Section 12.1), and Change Dump (Section 13.1). A Capability List must only contain one entry per capability.

Capabilities that are conveyed in the same Capability List uniformly apply to the set of resources covered by that Capability List. For example, if a Capability List enumerates a Resource List, a Resource Dump, and a Change List, then a given resource that appears in a Resource List must also appear in a Resource Dump, and changes to the resource must be conveyed in the Change List.

The Capability List is based on the <urlset> format. It has the <urlset> root element and the following structure:

- The mandatory <rs:md> child element of <urlset> must have a capability attribute that has a value of capabilitylist.
- A mandatory <rs:ln> child element of <urlset> with the relation type up points to the Source Description document that enumerates all Capability Lists offered by the Source.
- A recommended <rs:ln> child element of <urlset> with the relation type describedby points to a document that provides information about the set of resources covered by the Capability List.
- One <url> child element of <urlset> per capability offered by the Source. This element does not have attributes, but uses child elements to convey information about the capabilities. The <url> element has the following child elements:
    - A mandatory <loc> child element provides the URI of the respective capability document.
    - A mandatory <rs:md> child element must have a capability attribute to convey the type of the respective capability.

The <lastmod> elements should be omitted from the Capability List unless the Source updates the Capability List every time it updates one of the capability documents.

Example 13 shows a Capability List where the Source offers four capabilities: a Resource List, a Resource Dump, a Change List, and a Change Dump. A Destination cannot determine from the Capability List whether a Source provides, for example, a Resource List Index or a single Resource List. The capability document must be downloaded to make this determination: a document with a <sitemapindex> root element is an index, a document with a <urlset> root element is not.

```
<?xml version="1.0" encoding="UTF-8"?>
<urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9"
        xmlns:rs="http://www.openarchives.org/rs/terms/">
  <rs:ln rel="describedby"
         href="http://example.com/info_about_set1_of_resources.xml"/>
  <rs:ln rel="up"
         href="http://example.com/resourcesync_description.xml"/>
  <rs:md capability="capabilitylist"/>
  <url>
      <loc>http://example.com/dataset1/resourcelist.xml</loc>
      <rs:md capability="resourcelist"/>
  </url>
  <url>
      <loc>http://example.com/dataset1/resourcedump.xml</loc>
      <rs:md capability="resourcedump"/>
  </url>
  <url>
      <loc>http://example.com/dataset1/changelist.xml</loc>
      <rs:md capability="changelist"/>
  </url>
  <url>
      <loc>http://example.com/dataset1/changedump.xml</loc>
      <rs:md capability="changedump"/>
  </url>
</urlset>
```

Example 13: A Capability List

ResourceSync defines only a small number of capabilities, and enumerating those does not approach the limits of a single Capability List. Extensions or revisions of this specification may introduce the use

of Capability List Indexes, but Sources should not generate such structures for the features introduced in this version of the ResourceSync specification.

10 Describing Resources

A Source may publish a description of the resources it makes available for synchronization. This information enables a Destination to make an initial copy of some or all of those resources, or to update a local copy to remain synchronized with changes.

10.1 Resource List

A Resource List is introduced to list and describe the resources that a Source makes available for synchronization. It presents a snapshot of a Source's resources at a particular point in time.

A Resource List is based on the `<urlset>` document format introduced by the Sitemap protocol. It has the `<urlset>` root element and the following structure:

- The mandatory `<rs:md>` child element of `<urlset>` must have a `capability` attribute that has a value of `resourcelist`. It must also have an `at` attribute that conveys the datetime at which the process of taking a snapshot of resources for their inclusion in the Resource List started, and it may have a `completed` attribute that conveys the datetime at which that process completed.
- A mandatory `<rs:ln>` child element of `<urlset>` points to the Capability List with the relation type `up`.
- In case a Resource List Index (see Section 10.2) exists, a recommended `<rs:ln>` child element of `<urlset>` points to it with the relation type `index`.
- One `<url>` child element of `<urlset>` should be included for each resource. This element does not have attributes, but uses child elements to convey information about the resource. The `<url>` element has the following child elements:
    - A mandatory `<loc>` child element provides the URI of the resource.
    - Optional `<lastmod>` and `<changefreq>` child element with semantics as described in Section 7.
    - An optional `<rs:md>` child element provides further metadata about the resource. It may have the attributes as described in Section 7.
    - Optional `<rs:ln>` child elements link to related resources as described in Section 7, and detailed in Section 14.

Example 14 shows a Resource List with two resources. The `at` attribute allows a Destination to determine that neither of the listed resources have undergone a change between their respective last modification datetimes, `2013-01-02T13:00:00Z` and `2013-01-02T14:00:00Z`, and the datetime that is the value of the `at` attribute, `2013-01-03T09:00:00Z`.

```
<?xml version="1.0" encoding="UTF-8"?>
<urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9"
        xmlns:rs="http://www.openarchives.org/rs/terms/">
  <rs:ln rel="up"
         href="http://example.com/dataset1/capabilitylist.xml"/>
  <rs:md capability="resourcelist"
         at="2013-01-03T09:00:00Z"
         completed="2013-01-03T09:01:00Z"/>
  <url>
      <loc>http://example.com/res1</loc>
      <lastmod>2013-01-02T13:00:00Z</lastmod>
      <rs:md hash="md5:1584abdf8ebdc9802ac0c6a7402c03b6"
             length="8876"
             type="text/html"/>
  </url>
  <url>
      <loc>http://example.com/res2</loc>
      <lastmod>2013-01-02T14:00:00Z</lastmod>
      <rs:md hash="md5:1e0d5cb8ef6ba40c99b14c0237be735e
                   sha-
256:854f61290e2e197a11bc91063afce22e43f8ccc655237050ace766adc68dc784"
             length="14599"
             type="application/pdf"/>
  </url>
</urlset>
```

Example 14: A Resource List

10.2 Resource List Index

The ResourceSync framework adopts the community-defined limits for publishing documents of the `<urlset>` format and introduces a **Resource List Index** for grouping multiple Resource Lists. The union of the Resource Lists referred to in the Resource List Index represents the entire set of resources that a Source makes available for synchronization. This set of resources, regardless of whether it is conveyed in a single Resource List or in multiple Resource Lists via a Resource List Index, represents the state of the Source's data at a point in time.

A Resource List Index is based on the `<sitemapindex>` document format introduced by the Sitemap protocol. It has the `<sitemapindex>` root element and the following structure:

- The mandatory `<rs:md>` child element of `<sitemapindex>` must have a `capability` attribute that has a value of `resourcelist`. It must also have an `at` attribute that conveys the datetime at which the process of taking a snapshot of resources for their inclusion in the Resource List Index started, and it may have a `completed` attribute that conveys the datetime at which that process completed.
- A mandatory `<rs:ln>` child element of `<sitemapindex>` points to the Capability List with the relation type `up`.
- One `<sitemap>` child element of `<sitemapindex>` should be included for each Resource List. This element does not have attributes, but uses child elements to convey information about the Resource List. The `<sitemap>` element has the following child elements:
  - A mandatory `<loc>` child element provides the URI of the Resource List.
  - An optional `<lastmod>` child element with semantics as described in Section 7.
  - An optional `<rs:md>` child element with an `at` attribute and possibly a `completed` attribute to convey the datetime at which the process of taking a snapshot of resources for their inclusion in the Resource List respectively started and ended.

The Destination can determine whether it has reached a Resource List or a Resource List Index based on whether the root element is `<urlset>` or `<sitemapindex>` respectively. A Resource List Index that points to three Resource Lists is shown in Example 15.

```
<?xml version="1.0" encoding="UTF-8"?>
<sitemapindex xmlns="http://www.sitemaps.org/schemas/sitemap/0.9"
              xmlns:rs="http://www.openarchives.org/rs/terms/">
  <rs:ln rel="up"
         href="http://example.com/dataset1/capabilitylist.xml"/>
  <rs:md capability="resourcelist"
         at="2013-01-03T09:00:00Z"
         completed="2013-01-03T09:10:00Z"/>
  <sitemap>
      <loc>http://example.com/resourcelist1.xml</loc>
      <rs:md at="2013-01-03T09:00:00Z"/>
  </sitemap>
  <sitemap>
      <loc>http://example.com/resourcelist2.xml</loc>
      <rs:md at="2013-01-03T09:03:00Z"/>
  </sitemap>
  <sitemap>
      <loc>http://example.com/resourcelist3.xml</loc>
      <rs:md at="2013-01-03T09:07:00Z"/>
  </sitemap>
</sitemapindex>
```

Example 15: A Resource List Index

Example 16 shows the content of the Resource List identified by the URI `http://example.com/resourcelist1.xml`. Structurally, it is identical to the Resource List shown in Example 14 but it contains an additional `<rs:ln>` child element of `<urlset>` that provides a navigational link with the relation type `index` to the parent Resource List Index shown in Example 15. This link is meant to ease navigation for Destinations and their adoption is therefore recommended.

```
<?xml version="1.0" encoding="UTF-8"?>
<urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9"
        xmlns:rs="http://www.openarchives.org/rs/terms/">
  <rs:ln rel="up"
         href="http://example.com/dataset1/capabilitylist.xml"/>
  <rs:ln rel="index"
         href="http://example.com/dataset1/resourcelist-index.xml"/>
  <rs:md capability="resourcelist"
         at="2013-01-03T09:00:00Z"/>
  <url>
      <loc>http://example.com/res3</loc>
      <lastmod>2013-01-02T13:00:00Z</lastmod>
      <rs:md hash="md5:1584abdf8ebdc9802ac0c6a7402c8753"
             length="4385"
             type="application/pdf"/>
  </url>
  <url>
      <loc>http://example.com/res4</loc>
      <lastmod>2013-01-02T14:00:00Z</lastmod>
      <rs:md hash="md5:4556abdf8ebdc9802ac0c6a7402c9881"
             length="883"
             type="image/png"/>
  </url>
</urlset>
```

Example 16: A Resource List with a navigational link to its parent Resource List Index

11 Packaging Resources

In order to provide Destinations with an efficient way to copy a Source's data using a small number of HTTP requests, a Source may provide packaged bitstreams for its resources.

11.1 Resource Dump

A Source may publish a **Resource Dump**, which provides links to packages of the resources' bitstreams. The Resource Dump represents the Source's state at a point in time. It may be used to transfer resources from the Source in bulk, rather than the Destination having to make many separate requests.

The ResourceSync framework specifies the use of the ZIP file format as the packaging format. Communities may define their own packaging format. A Resource Dump should only point to packages of the same format.

A Resource Dump is based on the `<urlset>` document format introduced by the Sitemap protocol. It has the `<urlset>` root element and the following structure:

- The mandatory `<rs:md>` child element of `<urlset>` must have a `capability` attribute that has a value of `resourcedump`. It must also have an `at` attribute that conveys the datetime at which the process of taking a snapshot of resources for their inclusion in the Resource Dump started, and it may have a `completed` attribute that conveys the datetime at which that process completed.
- A mandatory `<rs:ln>` child element of `<urlset>` points to the Capability List with the relation type `up`.
- In case a Resource Dump Index (see Section 11.2) exists, a recommended `<rs:ln>` child element of `<urlset>` points to it with the relation type `index`.
- One `<url>` child element of `<urlset>` should be included for each bitstream package. This element does not have attributes, but uses child elements to convey information about the package. The `<url>` element has the following child elements:
    - A mandatory `<loc>` child element provides the URI of the package.
    - An optional `<lastmod>` child element with semantics as described in Section 7.
    - A recommended `<rs:md>` child element to convey the Media Type and the length of the package using the `type` and `length` attribute, respectively. It may also have an `at` attribute and possibly a `completed` attribute to convey the datetime at which the process of taking a snapshot of resources for their inclusion in the package respectively started and ended. The child element may further have attributes such as `hash`, as described in Section 7.
    - An optional `<rs:ln>` child element with the relation type `contents` that points to the Resource Dump Manifest associated with the bitstream package.

Example 17 shows a Resource Dump that points to three ZIP files. Included in each <url> element is a pointer to the Resource Dump Manifest associated with the package. While this pointer is optional and intended for the Destination's convenience, if provided, the Source needs to ensure that the referred Manifest corresponds with the Manifest included in the bitstream package.

```
<?xml version="1.0" encoding="UTF-8"?>
<urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9"
        xmlns:rs="http://www.openarchives.org/rs/terms/">
  <rs:ln rel="up"
         href="http://example.com/dataset1/capabilitylist.xml"/>
  <rs:md capability="resourcedump"
         at="2013-01-03T09:00:00Z"
         completed="2013-01-03T09:04:00Z"/>
  <url>
      <loc>http://example.com/resourcedump-part1.zip</loc>
      <rs:md type="application/zip"
             length="4765"
             at="2013-01-03T09:00:00Z"
             completed="2013-01-03T09:02:00Z"/>
      <rs:ln rel="contents"
             href="http://example.com/resourcedump_manifest-part1.xml"
             type="application/xml"/>
  </url>
  <url>
      <loc>http://example.com/resourcedump-part2.zip</loc>
      <rs:md type="application/zip"
             length="9875"
             at="2013-01-03T09:01:00Z"
             completed="2013-01-03T09:03:00Z"/>
      <rs:ln rel="contents"
             href="http://example.com/resourcedump_manifest-part2.xml"
             type="application/xml"/>
  </url>
  <url>
      <loc>http://example.com/resourcedump-part3.zip</loc>
      <rs:md type="application/zip"
             length="2298"
             at="2013-01-03T09:03:00Z"
             completed="2013-01-03T09:04:00Z"/>
      <rs:ln rel="contents"
             href="http://example.com/resourcedump_manifest-part3.xml"
             type="application/xml"/>
  </url>
</urlset>
```

Example 17: A Resource Dump

If a Source needs to or chooses to publish multiple Resource Dumps, it must group them using a Resource Dump Index, in a manner that is similar to what was described in Section 11.2.

11.2 Resource Dump Manifest

Each ZIP package referred to from a Resource Dump must contain a **Resource Dump Manifest** file that describes the package's constituent bitstreams. The file must be named `manifest.xml` and must be located at the top level of the ZIP package.

The Resource Dump Manifest is based on the `<urlset>` format. It has the `<urlset>` root element and the following structure:

- The mandatory `<rs:md>` child element of `<urlset>` must have a `capability` attribute with a value of `resourcedump-manifest`. It must also have an `at` attribute that conveys the datetime at which the process of taking a snapshot of resources for their inclusion in the ZIP package started, and it may have a `completed` attribute that conveys the datetime at which that process completed.
- A mandatory `<rs:ln>` child element of `<urlset>` points to the Capability List with the relation type `up`.

- One `<url>` child element of `<urlset>` per bitstream. This element does not have attributes, but uses child elements to convey information about the bitstream. The `<url>` element has the following child elements:
  - A mandatory `<loc>` child element provides the URI that the Source associates with the bitstream.
  - Optional `<lastmod>` and `<changefreq>` child elements with semantics as described in Section 7.
  - A mandatory `<rs:md>` child element must have a `path` attribute to convey the location of the bitstream within the package. The value of the attribute is relative to root of the package and it is expressed with a leading slash (/). The use of the `type` attribute in the `<rs:md>` element is recommended to help Destinations determine the Media Type of the bitstream. The `<rs:md>` element may further have the attributes `hash` and `length`, as described in Section 7.
  - Optional `<rs:ln>` child elements link to related resources as described in Section 7, and detailed in Section 14.

Example 18 shows a Resource Dump Manifest for a ZIP file that contains two bitstreams.

```
<?xml version="1.0" encoding="UTF-8"?>
<urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9"
        xmlns:rs="http://www.openarchives.org/rs/terms/">
  <rs:ln rel="up"
         href="http://example.com/dataset1/capabilitylist.xml"/>
  <rs:md capability="resourcedump-manifest"
         at="2013-01-03T09:00:00Z"
         completed="2013-01-03T09:02:00Z"/>
  <url>
      <loc>http://example.com/res1</loc>
      <lastmod>2013-01-02T13:00:00Z</lastmod>
      <rs:md hash="md5:1584abdf8ebdc9802ac0c6a7402c03b6"
             length="8876"
             type="text/html"
             path="/resources/res1"/>
  </url>
  <url>
      <loc>http://example.com/res2</loc>
      <lastmod>2013-01-02T14:00:00Z</lastmod>
      <rs:md hash="md5:1e0d5cb8ef6ba40c99b14c0237be735e
                   sha-256:854f61290e2e197a11bc91063afce22e43f8ccc655237050ace766adc68dc784"
             length="14599"
             type="application/pdf"
             path="/resources/res2"/>
  </url>
</urlset>
```

Example 18: A Resource Dump Manifest

12 Describing Changes

A Source may publish a record of the changes to its resources. This enables Destinations to efficiently learn about those changes and hence to synchronize incrementally.

12.1 Change List

A **Change List** is a document that contains a description of changes to a Source's resources. It is up to the Source to determine the publication frequency of Change Lists, as well as the temporal interval

they cover. For example, a Source may choose to publish a fixed number of changes per Change List, or all the changes in a period of fixed length, such as an hour, a day, or a week. All entries in a Change List must be provided in forward chronological order: the least recently changed resource must be listed at the beginning of the Change List, while the most recently changed resource must be listed at the end of the document. If a resource underwent multiple changes in the period covered by a Change List, then it will be listed multiple times, once per change.

A Change List is based on the `<urlset>` document format introduced by the Sitemap protocol. It has the `<urlset>` root element and the following structure:

- The mandatory `<rs:md>` child element of `<urlset>` must have a `capability` attribute that has a value of `changelist`. It also has the mandatory `from` and the optional `until` attributes to convey the temporal interval covered by the Change List. Details about the semantics of these attributes are provided below.
- A mandatory `<rs:ln>` child element of `<urlset>` points to the Capability List with the relation type `up`.
- In case a Change List Index (see Section 12.2) exists, a recommended `<rs:ln>` child element of `<urlset>` points to it with the relation type `index`.
- One `<url>` child element of `<urlset>` should be included for each resource change. This element does not have attributes, but uses child elements to convey information about the changed resource. The `<url>` element has the following child elements:
    - A mandatory `<loc>` child element provides the URI of the changed resource.
    - A mandatory `<lastmod>` and an optional `<changefreq>` child element with semantics as described in Section 7.
    - A mandatory `<rs:md>` child element must have the attribute `change` to convey the nature of the change. Its value is `created`, `updated`, or `deleted`. It may further have attributes `hash`, `length`, and `type`, as described in Section 7.
    - Optional `<rs:ln>` child elements link to related resources as described in Section 7 and detailed in Section 14.

The temporal interval covered by a Change List is conveyed by means of the `from` and `until` attributes of the `<rs:md>` child element of the `<urlset>` root element. The `from` attribute indicates that the Change List includes all changes that occurred to the set of resources at the Source since the datetime expressed as the value of the attribute. If it exists, the `until` attribute indicates that the Change List includes all changes that occurred to the set of resources at the Source up until the datetime expressed as the value of the attribute. Its use is optional for Change Lists:

- When a document carries the `until` attribute, this indicates that the document will not be updated anymore; the Change List is closed. When a Destination has finished processing a closed Change List, it should consult the enclosing Change List Index (following the link with the `index` relation type), if applicable, or the Capability List (following the link with the `up` relation type) to determine the URI of the Change List that reports the changes that occurred after the datetime expressed as the closed Change List's `until` value.
- When a document does not carry the `until` attribute, this indicates that the document will be updated with further changes; the Change List remains open. A Destination should continue to poll an open Change List to learn about further changes. It does not need to consult the enclosing Change List Index, if applicable, nor the Capability List.

The `from` and `until` attributes help a Destination to determine whether it has or has not fully processed a Change List. The forward chronological order of changes in a Change List, the datetime of a resource change, and the URI of a changed resource help the Destination to determine the first unprocessed change in a not fully processed Change List. The Destination should start processing

there; it can retrieve a representation of a changed resource by dereferencing its URI provided in the `<loc>` child element of the `<url>` element that conveys the change. In order for the determination of the first unprocessed change to be accurate, the combination of the URI of a changed resource and the datetime of its change should be unique. Hence, a Source should provide change datetime values at a sufficiently fine granularity.

Example 19 shows a Change List that indicates that four resource changes occurred since `2013-01-03T00:00:00Z`: one creation, two updates, and one deletion. One resource underwent two of these changes and hence is listed twice. The Change List has no `until` attribute, which indicates that it will report further changes; a Destination should keep polling this Change List.

```
<?xml version="1.0" encoding="UTF-8"?>
<urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9"
        xmlns:rs="http://www.openarchives.org/rs/terms/">
  <rs:ln rel="up"
         href="http://example.com/dataset1/capabilitylist.xml"/>
  <rs:md capability="changelist"
         from="2013-01-03T00:00:00Z"/>
  <url>
      <loc>http://example.com/res1.html</loc>
      <lastmod>2013-01-03T11:00:00Z</lastmod>
      <rs:md change="created"/>
  </url>
  <url>
      <loc>http://example.com/res2.pdf</loc>
      <lastmod>2013-01-03T13:00:00Z</lastmod>
      <rs:md change="updated"/>
  </url>
  <url>
      <loc>http://example.com/res3.tiff</loc>
      <lastmod>2013-01-03T18:00:00Z</lastmod>
      <rs:md change="deleted"/>
  </url>
  <url>
      <loc>http://example.com/res2.pdf</loc>
      <lastmod>2013-01-03T21:00:00Z</lastmod>
      <rs:md change="updated"/>
  </url>
</urlset>
```

Example 19: An open Change List describing four resource changes

12.2 Change List Index

If a Source needs to publish multiple Change Lists, it must group them in a Change List Index. A Change List Index must enumerate Change Lists in forward chronological order.

A Change List Index is based on the `<sitemapindex>` document format introduced by the Sitemap protocol. It has the `<sitemapindex>` root element and the following structure:

- The mandatory `<rs:md>` child element of `<sitemapindex>` must have a `capability` attribute that has a value of `changelist`. It also has the mandatory `from` and the optional `until` attributes to convey the temporal interval covered by the Change List Index. The semantics of these attributes are as explained in detail for Change Lists (see Section 12.1).
- A mandatory `<rs:ln>` child element of `<sitemapindex>` points to the Capability List with the relation type `up`.

- One <sitemap> child element of <sitemapindex> should be included for each Change List. This element does not have attributes, but uses child elements to convey information about the Change List. The <sitemap> element has the following child elements:
  - A mandatory <loc> child element provides the URI of the Change List.
  - An optional <lastmod> child element with semantics as described in Section 7.
  - A recommended <rs:md> child element with the from and possibly until attributes to convey the temporal interval covered by the Change List. The use of the until is as follows:
    - If the Change List is closed the use of <until> is required.
    - If the Change List is open <until> must not be provided.

The Destination should determine whether it has reached a Change List or a Change List Index based on whether the root element is <urlset> or <sitemapindex> respectively.

A Change List Index that points to three Change Lists is shown in Example 20. Two of those Change Lists are closed, as indicated by the provision of <lastmod>, and one is open, as indicated by its absence. The closed Change List http://example.com/20130102-changelist.xml is shown in Example 21. Note that the value for <lastmod> for this Change List in the Change List Index is the same as the value of the until attribute in the Change List: 2013-01-02T23:59:59Z. The open Change List could be the one shown in Example 19, in which case that list would have an additional link with an index relation type pointing to the Change List Index.

```
<?xml version="1.0" encoding="UTF-8"?>
<sitemapindex xmlns="http://www.sitemaps.org/schemas/sitemap/0.9"
        xmlns:rs="http://www.openarchives.org/rs/terms/">
  <rs:ln rel="up"
         href="http://example.com/dataset1/capabilitylist.xml"/>
  <rs:md capability="changelist"
         from="2013-01-01T00:00:00Z"/>
  <sitemap>
      <loc>http://example.com/20130101-changelist.xml</loc>
      <rs:md from="2013-01-01T00:00:00Z"
             until="2013-01-02T00:00:00Z"/>
  </sitemap>
  <sitemap>
      <loc>http://example.com/20130102-changelist.xml</loc>
      <rs:md from="2013-01-02T00:00:00Z"
             until="2013-01-03T00:00:00Z"/>
  </sitemap>
  <sitemap>
      <loc>http://example.com/20130103-changelist.xml</loc>
      <rs:md from="2013-01-03T00:00:00Z"/>
  </sitemap>
</sitemapindex>
```

Example 20: A Change List Index

```
<?xml version="1.0" encoding="UTF-8"?>
<urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9"
        xmlns:rs="http://www.openarchives.org/rs/terms/">
  <rs:ln rel="up"
         href="http://example.com/dataset1/capabilitylist.xml"/>
  <rs:ln rel="index"
         href="http://example.com/dataset1/changelist.xml"/>
  <rs:md capability="changelist"
         from="2013-01-02T00:00:00Z"
         until="2013-01-03T00:00:00Z"/>
  <url>
      <loc>http://example.com/res7.html</loc>
      <lastmod>2013-01-02T12:00:00Z</lastmod>
      <rs:md change="created"/>
  </url>
  <url>
      <loc>http://example.com/res9.pdf</loc>
      <lastmod>2013-01-02T13:00:00Z</lastmod>
      <rs:md change="updated"/>
  </url>
  <url>
      <loc>http://example.com/res5.tiff</loc>
      <lastmod>2013-01-02T19:00:00Z</lastmod>
      <rs:md change="deleted"/>
  </url>
  <url>
      <loc>http://example.com/res7.html</loc>
      <lastmod>2013-01-02T20:00:00Z</lastmod>
      <rs:md change="updated"/>
  </url>
</urlset>
```

Example 21: A closed Change List pointing back to its Index

13 Packaging Changes

In order to reduce the number of requests required to obtain resource changes, a Source may provide packaged bitstreams for changed resources.

13.1 Change Dump

To make content changes available for download, a Source may publish **Change Dumps**. A Change Dump is a document that points to packages containing bitstreams for the Source's changed resources. The ResourceSync framework specifies the use of the ZIP file format as the packaging format. Communities may define their own packaging format. A Change Dump should only point to packages of the same format.

It is up to the Source to determine the publication frequency of these packages, as well as the temporal interval they cover. For example, a Source may choose to publish a fixed number of changes per package, or all the changes in a period of fixed length, such as an hour, a day, or a week. If a resource underwent multiple changes in the period covered by a package, then the package will contain multiple bitstreams for the resource, one per change. As new packages are published, new entries are added to the Change Dump that points at them. All entries in a Change Dump should be provided in forward chronological order: the least recently published package listed at the beginning of the Change Dump, the most recent package listed at the end of the document.

A Change Dump is based on the `<urlset>` document format introduced by the Sitemap protocol. It has the `<urlset>` root element and the following structure:

- The mandatory <rs:md> child element of <urlset> must have a capability attribute that has a value of changedump. It also has the mandatory from and the optional until attributes to convey the temporal interval covered by the Change Dump. The semantics of these attributes are as explained in detail for Change Lists (see Section 12.1).
- A mandatory <rs:ln> child element of <urlset> points to the Capability List with the relation type up.
- In case a Change Dump Index (see Section 13.2) exists, a recommended <rs:ln> child element of <urlset> points to it with the relation type index.
- One <url> child element of <urlset> should be included for each bitstream package. This element does not have attributes, but uses child elements to convey information about the package. The <url> element has the following child elements:
    - A mandatory <loc> child element provides the URI of the package.
    - An optional <lastmod> child element with semantics as described in Section 7.
    - A recommended <rs:md> child element to convey the Media Type and the length of the package using the type and length attribute, respectively. It may also have the from and until attributes to convey the temporal interval covered by the package. The child element may further have attributes such as hash, as described in Section 7.
    - An optional <rs:ln> child element with the relation type contents that points to a copy of the Change Dump Manifest associated with each package.

Example 22 shows a Change Dump with pointers to three bitstream packages associated with changed resources. The absence of the until attribute indicates that further packages will be added. The example also includes within each <url> element a pointer to a copy of the Change Dump Manifest (see Section 13.2) associated with the package. This pointer is optional and intended to allow a Destination to determine whether the package should be downloaded. If such pointers are provided, the Source must ensure that the Manifest referred to matches the Manifest included in the bitstream package.

```
<?xml version="1.0" encoding="UTF-8"?>
<urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9"
        xmlns:rs="http://www.openarchives.org/rs/terms/">
  <rs:ln rel="up"
         href="http://example.com/dataset1/capabilitylist.xml"/>
  <rs:md capability="changedump"
         from="2013-01-01T00:00:00Z"/>
  <url>
      <loc>http://example.com/20130101-changedump.zip</loc>
      <lastmod>2013-01-01T23:59:59Z</lastmod>
      <rs:md type="application/zip"
             length="3109"
             from="2013-01-01T00:00:00Z"
             until="2013-01-02T00:00:00Z"/>
      <rs:ln rel="contents"
             href="http://example.com/20130101-changedump-manifest.xml"
             type="application/xml"/>
  </url>
  <url>
      <loc>http://example.com/20130102-changedump.zip</loc>
      <lastmod>2013-01-02T23:59:59Z</lastmod>
      <rs:md type="application/zip"
             length="6629"
             from="2013-01-02T00:00:00Z"
             until="2013-01-03T00:00:00Z"/>
      <rs:ln rel="contents"
             href="http://example.com/20130102-changedump-manifest.xml"
             type="application/xml"/>
  </url>
  <url>
      <loc>http://example.com/20130103-changedump.zip</loc>
      <lastmod>2013-01-03T23:59:59Z</lastmod>
      <rs:md type="application/zip"
             length="8124"
             from="2013-01-03T00:00:00Z"
             until="2013-01-04T00:00:00Z"/>
      <rs:ln rel="contents"
             href="http://example.com/20130103-changedump-manifest.xml"
             type="application/xml"/>
  </url>
</urlset>
```

Example 22: A Change Dump

If a Source needs to publish multiple Change Dumps, it must group them in a Change Dump Index, in a manner similar to what was described in Section 12.2.

13.2 Change Dump Manifest

Each ZIP package referred to from a Change Dump must contain a **Change Dump Manifest** file that describes the constituent bitstreams of the package. The file must be named `manifest.xml` and must be located at the top level of the ZIP package. All entries in a Change Dump Manifest must be provided in forward chronological order: the bitstream associated with the least recent resource change is listed first, and the bitstream associated with the most recent change is listed last.

The Change Dump Manifest is based on the `<urlset>` format. It has the `<urlset>` root element and the following structure:

- The mandatory `<rs:md>` child element of `<urlset>` must have a `capability` attribute that has a value of `changedump-manifest`. It also has the mandatory `from` and the optional `until` attributes to convey the temporal interval covered by the ZIP package in which the Change Dump Manifest is contained: bitstream for all resource changes that

occurred during the specified interval must be included in the ZIP package and the Change Dump Manifest must refer to each.

- A mandatory <rs:ln> child element of <urlset> points to the Capability List with the relation type up.
- One <url> child element of <urlset> should be included for each resource change, and hence for each bitstream. This element does not have attributes, but uses child elements to convey information about the resource change and its associated bitstream. The <url> element has the following child elements:
  - A mandatory <loc> child element provides the URI which the Source associates with the bitstream.
  - A mandatory <lastmod> and an optional <changefreq> child element with semantics as described in Section 7.
  - A mandatory <rs:md> child element must have a change attribute to convey the type of change the resource underwent. The change attribute value may be created, updated, or deleted. Except in the case of change="deleted", the element must also have a path attribute to convey the location of the bitstream within the ZIP package. The path is relative to root of the package and it is expressed with a leading slash (/). The use of the type attribute in the <rs:md> element is strongly recommended to help Destinations determine the Media Type of the bitstream. The <rs:md> element may further have the attributes hash and length, as described in Section 7.
  - Optional <rs:ln> child elements link to related resources as described in Section 7 and detailed in Section 14.

Example 23 shows the Change Dump Manifest associated with the second entry in the Resource Dump from Example 22. The Manifest must be named manifest.xml at the top level of the ZIP package. A copy of the Manifest may also be provided at a location indicated by an optional <rs:ln> element with the relation type contents in the Change Dump, http://example.com/20130102-changedump-manifest.xml in Example 22. The Manifest covers the same changes as conveyed in the closed Change List of Example 21. The resource http://example.com/res7.html is listed twice, once because it was created, and once because it was updated. Both entries have the same URI. The ZIP package in which this Manifest is contained has two bitstreams for this resource, available at different paths in the package.

```
<?xml version="1.0" encoding="UTF-8"?>
<urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9"
        xmlns:rs="http://www.openarchives.org/rs/terms/">
  <rs:ln rel="up"
         href="http://example.com/dataset1/capabilitylist.xml"/>
  <rs:md capability="changedump-manifest"
         from="2013-01-02T00:00:00Z"
         until="2013-01-03T00:00:00Z"/>
  <url>
      <loc>http://example.com/res7.html</loc>
      <lastmod>2013-01-02T12:00:00Z</lastmod>
      <rs:md change="created"
             hash="md5:1c1b0e264fa9b7e1e9aa6f9db8d6362b"
             length="4339"
             type="text/html"
             path="/changes/res7.html"/>
  </url>
  <url>
      <loc>http://example.com/res9.pdf</loc>
      <lastmod>2013-01-02T13:00:00Z</lastmod>
      <rs:md change="updated"
             hash="md5:f906610c3d4aa745cb2b986f25b37c5a"
             length="38297"
             type="application/pdf"
             path="/changes/res9.pdf"/>
  </url>
  <url>
      <loc>http://example.com/res5.tiff</loc>
      <lastmod>2013-01-02T19:00:00Z</lastmod>
      <rs:md change="deleted"/>
  </url>
  <url>
      <loc>http://example.com/res7.html</loc>
      <lastmod>2013-01-02T20:00:00Z</lastmod>
            <rs:md change="updated"
             hash="md5:0988647082c8bc51778894a48ec3b576"
             length="5426"
             type="text/html"
             path="/changes/res7-v2.html"/>
  </url>
</urlset>
```

Example 23: A Change Dump Manifest

14 Linking to Related Resources

14.1 Overview

In order to facilitate alternative approaches to obtain content for a resource that is subject to synchronization or to provide additional information about it, a Source may provide links from that resource to related resources. Such links may occur in Resource Lists, Resource Dump Manifests, Change Lists, and Change Dump Manifests. The following cases are considered and detailed (in examples of Change Lists) in the remainder of this section:

- The Source makes the resource content available at a mirror location.
- The Source provides alternate representations for the resource.
- The Source details the difference between the current and the previous version of the resource.

- The Source makes a resource as well as metadata about the resource available for synchronization.
- The Source provides access to prior versions of the resource.
- The Source provides collection membership information about the resource.
- A Destination republishes a resource obtained from a Source, makes that republished resource available for synchronization (i.e. acts as a Source itself), and links to the original resource that it republished.

As always, the `<loc>` child element of `<url>` conveys the URI of the resource that is subject to synchronization. The related resource is provided by means of the `<rs:ln>` child element of `<url>`. The possible attributes for `<rs:ln>` as well as the link relation types used to address the aforementioned use cases are detailed in Section 7. Links to meet needs other than the ones listed may be provided, and appropriate relation types may be selected from the IANA Link Relation Type Registry or expressed as URIs as specified in RFC 5988, Sec. 4.2.

In case a Destination is not able to adequately interpret the information conveyed in an `<rs:ln>` element, it should refrain from accessing the related resource and rather use the URI provided in `<loc>` to retrieve the resource.

14.2 Mirrored Content

In order to reduce the load on its primary access mechanism, a Source may convey one or mirror locations for a resource. An `<rs:ln>` element is introduced to express each mirror location for the resource. This element has the following attributes:

- A mandatory `rel` attribute with a value of `duplicate`.
- A mandatory `href` attribute that conveys the URI of the mirrored resource.
- An optional `pri` attribute to express a prioritization among multiple mirror locations, each expressed by means of an individual `<rs:ln>` element. The use of `pri` is detailed in Section 7.
- Other attributes are as described for the `<rs:ln>` child element of `<url>` in Section 7.

Example 24 shows how a Source conveys information about prioritized mirror locations for a resource. Since the three locations conveyed by `<rs:ln>` elements point to duplicates of the resource specified in `<loc>`, the values for each of the attributes of `<rs:md>` are expected to be identical for the resource and its mirrors. Hence, they should be omitted from the `<rs:ln>` elements. The last `<rs:ln>` element points to a mirror location where the resource is accessible via a protocol other than HTTP as can be seen from the URI scheme. Even though the resources are duplicates, their last modified datetimes may vary.

```
<?xml version="1.0" encoding="UTF-8"?>
<urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9"
        xmlns:rs="http://www.openarchives.org/rs/terms/">
  <rs:ln rel="up"
         href="http://example.com/dataset1/capabilitylist.xml"/>
  <rs:md capability="changelist"
         from="2013-01-03T00:00:00Z"/>
  <url>
      <loc>http://example.com/res1</loc>
      <lastmod>2013-01-03T18:00:00Z</lastmod>
      <rs:md change="updated"
             hash="md5:1584abdf8ebdc9802ac0c6a7402c03b6"
             length="8876"
             type="text/html"/>
      <rs:ln rel="duplicate"
             pri="1"
             href="http://mirror1.example.com/res1"
             modified="2013-01-03T18:00:00Z"/>
      <rs:ln rel="duplicate"
             pri="2"
             href="http://mirror2.example.com/res1"
             modified="2013-01-03T18:00:00Z"/>
      <rs:ln rel="duplicate"
             pri="3"
             href="gsiftp://gridftp.example.com/res1"
             modified="2013-01-03T18:00:00Z"/>
  </url>
</urlset>
```

Example 24: Mirrored content

14.3 Alternate Representations

A resource may have multiple representations available from different URIs. A resource may, for example, be identified by a generic URI such as `http://example.com/res1`. After performing content negotiation with the server, a client may, for example, obtain the resource's HTML representation available from the specific URI `http://example.com/res1.html`. Another client may ask for and retrieve the PDF representation of the resource from the specific URI `http://example.com/res1.pdf`. Which representation a client obtains, can, amongst others, depend on its preferences in terms of Media Type and language, its geographical location, and its device type.

A Source may express that a resource is subject to synchronization by conveying its generic URI in `<loc>`. In this case, per alternate representation that the Source wants to advertise, an `<rs:ln>` element is introduced. This element has the following attributes:

- A mandatory `rel` attribute with a value of `alternate`.
- A mandatory `href` attribute that conveys the specific URI of the alternate representation of the resource.
- A recommended `type` attribute that conveys the Media Type of the alternate representation.
- Other attributes are as described for the `<rs:ln>` child element of `<url>` in Section 7.

Cases exist in which there is no generic URI for a resource, only specific URIs. This may occur, for example, when a resource has different representations available for different devices. In this case the URI in `<loc>` will be a specific URI, and `<rs:ln>` elements with an `alternate` relation type are still used to refer to alternate representations available from other specific URIs.

Example 25 shows how to promote a generic URI in <loc> while also pointing to alternate representations available from specific URIs, for example, through content negotiation.

```
<?xml version="1.0" encoding="UTF-8"?>
<urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9"
        xmlns:rs="http://www.openarchives.org/rs/terms/">
  <rs:ln rel="up"
         href="http://example.com/dataset1/capabilitylist.xml"/>
  <rs:md capability="changelist"
         from="2013-01-03T11:00:00Z"/>
  <url>
      <loc>http://example.com/res1</loc>
      <lastmod>2013-01-03T18:00:00Z</lastmod>
      <rs:md change="updated"/>
      <rs:ln rel="alternate"
             href="http://example.com/res1.html"
             modified="2013-01-03T18:00:00Z"
             type="text/html"/>
      <rs:ln rel="alternate"
             href="http://example.com/res1.pdf"
             modified="2013-01-03T18:00:00Z"
             type="application/pdf"/>
  </url>
</urlset>
```

Example 25: Generic URI and alternates with specific URIs

In cases where a particular representation is considered the subject of synchronization, its specific URI is provided in <loc>. The associated generic URI, if one exists, may be provided using an <rs:ln> element. This element has the following attributes:

- A mandatory rel attribute with a value of canonical.
- A mandatory href attribute that conveys the generic URI associated with the specific URI provided in <loc>.
- Other attributes are as described for the <rs:ln> child element of <url> in Section 7.

This approach might be most appropriate for Resource Dump Manifests and Change Dump Manifests that describe bitstreams contained in a ZIP file.

Example 26 shows a Source promoting a specific URI in <loc> while also pointing to the resource's generic URI by means of an <rs:ln> element. Metadata pertaining to the representation available from that specific URI is conveyed by means of attributes of the <rs:md> element.

```
<?xml version="1.0" encoding="UTF-8"?>
<urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9"
        xmlns:rs="http://www.openarchives.org/rs/terms/">
  <rs:ln rel="up"
         href="http://example.com/dataset1/capabilitylist.xml"/>
  <rs:md capability="changelist"
         from="2013-01-03T00:00:00Z"/>
  <url>
      <loc>http://example.com/res1.html</loc>
      <lastmod>2013-01-03T18:00:00Z</lastmod>
      <rs:md change="updated"
             hash="md5:1584abdf8ebdc9802ac0c6a7402c03b6"
             length="8876"/>
      <rs:ln rel="canonical"
             href="http://example.com/res1"
             modified="2013-01-03T18:00:00Z"/>
  </url>
</urlset>
```

Example 26: Specific URI and alternate with generic URI

14.4 Patching Content

In order to increase the efficiency of updating a resource, a Source may make a description available of the changes that the resource underwent, in addition to the entire changed resource. Especially when frequent minor changes and/or changes to large resources are concerned, such an approach may be attractive. It will, however, require an unambiguous way to describe the changes in such a way that a Destination can construct the most recent version of the resource by appropriately patching the previous version with the description of the changes.

A Source may express that it makes a description of resource changes available by providing the URI of the resource in `<loc>`, as usual, and by introducing an `<rs:ln>` element with the following attributes:

- A mandatory `rel` attribute with a value of `http://www.openarchives.org/rs/terms/patch`.
- A mandatory `href` attribute that conveys the URI of the description of the resource changes.
- A mandatory `type` attribute that conveys the Media Type of the change description. That Media Type must be such that it allows the unambiguous application of the described changes to the previous version of the resource to construct the current one.
- Other attributes are as described for the `<rs:ln>` child element of `<url>` in Section 7.

Example 27 shows a Source that expresses changes that a JSON resource underwent expressed using the `application/json-patch` Media Type introduced in JSON Patch. It also shows the Source conveying changes to a large TIFF file using an experimental Media Type that may, for example, be described in a community specification. A Destination that does not understand the Media Type should ignore the description of changes and use the URI in `<loc>` to obtain the most recent version of the resource. Another example of a well-specified Media Type for expressing changes to XML document is `application/patch-ops-error+xml`, as specified in RFC 5261.

Expressing resource changes in this manner is only applicable to Change Lists (as in Example 27) and Change Dumps. When doing so for a Change Dump, the entry in the Change Dump Manifest must have the `path` attribute for the `<rs:ln>` element that points to the change description that is included in the content package.

```
<?xml version="1.0" encoding="UTF-8"?>
<urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9"
        xmlns:rs="http://www.openarchives.org/rs/terms/">
  <rs:ln rel="up"
         href="http://example.com/dataset1/capabilitylist.xml"/>
  <rs:md capability="changelist"
         from="2013-01-03T00:00:00Z"/>
  <url>
      <loc>http://example.com/res4</loc>
      <lastmod>2013-01-03T17:00:00Z</lastmod>
      <rs:md change="updated"
             hash="sha-256:f4OxZX_x_DFGFDgghgdfb6rtSx-iosjf6735432nklj"
             length="56778"
             type="application/json"/>
      <rs:ln rel="http://www.openarchives.org/rs/terms/patch"
             href="http://example.com/res4-json-patch"
             modified="2013-01-03T17:00:00Z"
             hash="sha-256:y66dER_t_HWEIKpesdkeb7rtSc-ippjf9823742opld"
             length="73"
             type="application/json-patch"/>
  </url>
  <url>
      <loc>http://example.com/res5-full.tiff</loc>
      <lastmod>2013-01-03T18:00:00Z</lastmod>
      <rs:md change="updated"
             hash="sha-256:f4OxZX_x_FO5LcGBSKHWXfwtSx-j1ncoSt3SABJtkGk"
             length="9788456778"
             type="image/tiff"/>
      <rs:ln rel="http://www.openarchives.org/rs/terms/patch"
             href="http://example.com/res5-diff"
             modified="2013-01-03T18:00:00Z"
             hash="sha-256:h986gT_t_87HTkjHYE76G558hY-jdfgy76t55sadJUYT"
             length="4533"
             type="application/x-tiff-diff"/>
  </url>
</urlset>
```

Example 27: A Change List with links to document that detail how to patch resources

14.5 Resources and Metadata about Resources

Cases exist where both resources and metadata about those resources must be synchronized. From the ResourceSync perspective, both the resource and the metadata about it are regarded as resources with distinct URIs that are subject to synchronization. As usual, each gets its distinct `<url>` block and each URI is conveyed in a `<loc>` child element of the respective block. If required, the inter-relationship between both resources is expressed by means of an `<rs:ln>` element with appropriate relation types added to each block. The `<rs:ln>` element has the following attributes:

- A mandatory `rel` attribute. When pointing from a resource to metadata that describes it, its value is `describedby`; when pointing from metadata to the resource described by the metadata, its value is `describes`.
- A mandatory `href` attribute. When pointing from a resource to metadata that describes it, its value is the URI of the metadata resource; when pointing from metadata to the resource described by it, the value is the URI of the described resource.
- Other attributes are as described for the `<rs:ln>` child element of `<url>` in Section 7.

Example 28 shows how a Source may express this inter-relationship between the two resources. Note that a Destination can use the metadata that describes a resource as a filtering mechanism to only synchronize with those resources that meet its metadata-based selection criteria. Note also in

the <url> element that conveys the metadata record update, the use of the link with a profile relation type [RFC 6906] to express the kind of metadata that is used to describe the resource, in this case, expressed by means of its XML Namespace. The link should provide a URI that supports the Destination in interpreting the metadata information. For example, it could refer to a namespace, an XML schema, or a description of MARC.

```
<?xml version="1.0" encoding="UTF-8"?>
<urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9"
        xmlns:rs="http://www.openarchives.org/rs/terms/">
  <rs:ln rel="up"
         href="http://example.com/dataset1/capabilitylist.xml"/>
  <rs:md capability="changelist"
         from="2013-01-03T00:00:00Z"/>
  <url>
      <loc>http://example.com/res2.pdf</loc>
      <lastmod>2013-01-03T18:00:00Z</lastmod>
      <rs:md change="updated"
             hash="md5:1584abdf8ebdc9802ac0c6a7402c03b6"
             length="8876"
             type="application/pdf"/>
      <rs:ln rel="describedby"
             href="http://example.com/res2_dublin-core_metadata.xml"
             modified="2013-01-01T12:00:00Z"
             type="application/xml"/>
  </url>
  <url>
      <loc>http://example.com/res2_dublin-core_metadata.xml</loc>
      <lastmod>2013-01-03T19:00:00Z</lastmod>
      <rs:md change="updated"
             type="application/xml"/>
      <rs:ln rel="describes"
             href="http://example.com/res2.pdf"
             modified="2013-01-03T18:00:00Z"
             hash="md5:1584abdf8ebdc9802ac0c6a7402c03b6"
             length="8876"
             type="application/pdf"/>
      <rs:ln rel="profile"
             href="http://purl.org/dc/elements/1.1/"/>
  </url>
</urlset>
```

Example 28: Linking between a resource and metadata about a resource in a Change List

14.6 Prior Versions of Resources

A Source may provide access to prior versions of a resource to allow Destinations to obtain a historical perspective, rather than just remaining synchronized with the most recent version. The approach to do so leverages a common resource versioning paradigm that consists of:

- The existence of a time-generic URI from which, at any moment in time, the current representation of the resource is available.
- The existence of a time-specific URI for each resource version, from which a specific temporal version of the resource is available.

When communicating about the resource, its time-generic URI is provided in <loc> and <lastmod> must be used to provide the resource's last modification time.

A first approach consists of conveying the time-specific URI of the resource that corresponds with the time of last modification, as given in the <lastmod> element. This is achieved by introducing a single <rs:ln> element with the following attributes:

- A mandatory rel attribute with a value of memento.

- A mandatory `href` attribute that conveys the time-specific URI of the resource that corresponds with the time of last modification. This URI allows a Destination to obtain that specific version during a catch-up operation, for example, because it had been offline, even if the resource has meanwhile changed again.
- Other attributes are as described for the `<rs:ln>` child element of `<url>` in Section 7.

A second approach consists of pointing to a TimeGate associated with the time-generic resource. A TimeGate supports negotiation in the datetime dimension, as introduced in the Memento protocol [Memento Internet Draft], to obtain a version of the resource as it existed at a specified moment in time. This allows the Destination to obtain the version that existed at the time of last modification by using the `<lastmod>` value in the datetime negotiation process, but also allows the Destination to obtain other versions by using different datetime values. A pointer to a TimeGate is introduced by using an `<rs:ln>` element with the following attributes:

- A mandatory `rel` attribute with a value of `timegate`.
- A mandatory `href` attribute that conveys the URI of TimeGate associated with the time-generic resource.
- The other attributes described for the `<rs:ln>` child element of `<url>` in Section 7 should not be used as they are meaningless for TimeGates.

A third approach consists of pointing to a TimeMap associated with the time-generic resource. A TimeMap, as introduced in the Memento protocol [Memento Internet Draft], enables Destinations to retrieve a comprehensive list of all time-specific resources known to a server. This allows Destinations to choose a particular version of a resource from that list. A pointer to a TimeMap is introduced by using an `<rs:ln>` element with the following attributes:

- A mandatory `rel` attribute with a value of `timemap`.
- A mandatory `href` attribute that conveys the URI of TimeMap associated with the time-generic resource.
- Other attributes are as described for the `<rs:ln>` child element of `<url>` in Section 7.

Example 29 shows a Change List with a link to a prior version of a resource, a link to a TimeGate, as well as a link to a TimeMap. Note that the values of the `hash, length,` and `type` attributes are identical between the `<rs:md>` child element and the `<rs:ln>` child element that points to the prior version.

```
<?xml version="1.0" encoding="UTF-8"?>
<urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9"
        xmlns:rs="http://www.openarchives.org/rs/terms/">
  <rs:ln rel="up"
         href="http://example.com/dataset1/capabilitylist.xml"/>
  <rs:md capability="changelist"
         from="2013-01-03T00:00:00Z"/>
  <url>
      <loc>http://example.com/res1</loc>
      <lastmod>2013-01-03T18:00:00Z</lastmod>
      <rs:md hash="md5:1584abdf8ebdc9802ac0c6a7402c03b6"
             length="8876"
             type="text/html"
             change="updated"/>
      <rs:ln rel="memento"
             href="http://example.com/20130103070000/res1"
             modified="2013-01-02T18:00:00Z"
             hash="md5:1584abdf8ebdc9802ac0c6a7402c03b6"
             length="8876"
             type="text/html"/>
      <rs:ln rel="timegate"
             href="http://example.com/timegate/http://example.com/res1"/>
      <rs:ln rel="timemap"
             href="http://example.com/timemap/http://example.com/res1"
             type="application/link-format"/>
  </url>
</urlset>
```

Example 29: Links to a resource version, and a Memento TimeGate and TimeMap

14.7 Collection Membership

A Source may express that a resource is a member of a collection such as an OAI-ORE Aggregation or an OAI-PMH Set. A Source may express collection membership of a resource that is subject to synchronization by providing the URI of that resource in `<loc>`, as usual, and by introducing an `<rs:ln>` element with the following attributes:

- A mandatory `rel` attribute with a value of `collection`.
- A mandatory `href` attribute that conveys the URI that identifies the collection.
- Other attributes are as described for the `<rs:ln>` child element of `<url>` in Section 7.

Example 30 shows a Change List with one resource that is a member of an OAI-ORE Aggregation.

```
<?xml version="1.0" encoding="UTF-8"?>
<urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9"
        xmlns:rs="http://www.openarchives.org/rs/terms/">
  <rs:ln rel="up"
         href="http://example.com/dataset1/capabilitylist.xml"/>
  <rs:md capability="changelist"
         from="2013-01-03T00:00:00Z"/>
  <url>
      <loc>http://example.com/res1</loc>
      <lastmod>2013-01-03T07:00:00Z</lastmod>
      <rs:md hash="md5:1584abdf8ebdc9802ac0c6a7402c03b6"
             length="8876"
             type="text/html"
             change="updated"/>
      <rs:ln rel="collection"
             href="http://example.com/aggregation/0601007"/>
  </url>
</urlset>
```

Example 30: A resource as a member of a collection

14.8 Republishing Resources

A special kind of Destination, henceforth called an aggregator, may retrieve content from a Source, republish it, and in its turn act as a Source for the republished content. In such an aggregator scenario, it may be important for a Destination that synchronizes with the aggregator to understand the provenance of the content and to be able to verify its accuracy with the original Source of the content. When communicating about a republished resource, the aggregator may provide such provenance information by introducing an `<rs:ln>` element with the following attributes:

- A mandatory `rel` attribute with a value of `via`.
- A mandatory `href` attribute that conveys the URI of the resource at the original Source.
- Other attributes are as described for the `<rs:ln>` child element of `<url>` in Section 7.

Example 31 shows a Change List in which a Source publishes information about a change to a single resource.

```
<?xml version="1.0" encoding="UTF-8"?>
<urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9"
        xmlns:rs="http://www.openarchives.org/rs/terms/">
  <rs:ln rel="up"
         href="http://example.com/dataset1/capabilitylist.xml"/>
  <rs:md capability="changelist"
         from="2013-01-03T00:00:00Z"/>
  <url>
      <loc>http://original.example.com/res1.html</loc>
      <lastmod>2013-01-03T07:00:00Z</lastmod>
      <rs:md hash="md5:1584abdf8ebdc9802ac0c6a7402c03b6"
             length="8876"
             type="text/html"
             change="updated"/>
  </url>
</urlset>
```

Example 31: An original Source publishes

Example 32 shows a primary aggregator's Change List that refers to the original Source's resource. It includes a link with the relation type `via` that has attributes such as `href` to convey information about the origin of the resource. This information corresponds with the data provided in the `<url>` block of the Change List shown in Example 31. For example, the value of the `href` attribute in Example 32 equals the value of the `<loc>` child element in the `<url>` block in Example 31.

```
<?xml version="1.0" encoding="UTF-8"?>
<urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9"
        xmlns:rs="http://www.openarchives.org/rs/terms/">
  <rs:ln rel="up"
         href="http://aggregator1.example.com/dataset1/capabilitylist.xml"/>
  <rs:md capability="changelist"
         from="2013-01-03T11:00:00Z"/>
  <url>
      <loc>http://aggregator1.example.com/res1.html</loc>
      <lastmod>2013-01-03T20:00:00Z</lastmod>
      <rs:md hash="md5:1584abdf8ebdc9802ac0c6a7402c03b6"
             length="8876"
             type="text/html"
             change="updated"/>
      <rs:ln rel="via"
             href="http://original.example.com/res1.html"
             modified="2013-01-03T07:00:00Z"
             hash="md5:1584abdf8ebdc9802ac0c6a7402c03b6"
             length="8876"
             type="text/html"/>
  </url>
</urlset>
```

Example 32: A primary aggregator republishes

If a secondary aggregator obtains the changed resource by consuming the Change List of the primary aggregator and republishes its Change List, a chain of aggregations is created. In this case each aggregator should maintain only the existing `via` link in order to convey information about the origin of the resource.

Example 33 shows the Change List of a secondary aggregator with information about the changed resource and the `via` link equal to Example 32. The data conveyed with the link corresponds to the data provided in the `<url>` block in Example 31.

```
<?xml version="1.0" encoding="UTF-8"?>
<urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9"
        xmlns:rs="http://www.openarchives.org/rs/terms/">
  <rs:ln rel="up"
         href="http://aggregator2.example.com/dataset1/capabilitylist.xml"/>
  <rs:md capability="changelist"
         from="2013-01-03T12:00:00Z"/>
  <url>
      <loc>http://aggregator2.example.com/res1.html</loc>
      <lastmod>2013-01-04T09:00:00Z</lastmod>
      <rs:md hash="md5:1584abdf8ebdc9802ac0c6a7402c03b6"
             length="8876"
             type="text/html"
             change="updated"/>
      <rs:ln rel="via"
             href="http://original.example.com/res1.html"
             modified="2013-01-03T07:00:00Z"
             hash="md5:1584abdf8ebdc9802ac0c6a7402c03b6"
             length="8876"
             type="text/html"/>
  </url>
</urlset>
```

Example 33: A second aggregator republishes

The values of the `at`, `completed`, `from`, and `until` attributes must always be expressed from the perspective of the Source that publishes the document that contains them. Hence it is possible that the `from` datetime of a Change List is more recent than the `<lastmod>` datetime of the original Source's resource described in the Change List, which is conveyed using an `<rs:ln>` link with the `via` relation type.

An aggregator should be cautious when inheriting links, other than the one with the `via` relation type, from a Source that precedes it in an aggregation chain. It should make sure that each such link remains appropriate from its own perspective and refrain from inheriting it when it is not. For example, a link with the relation type `collection` or `canonical` expressed by the original Source may not be appropriate in the context of the aggregator's copy, and hence should not be included in the description of the changed resource in the aggregator's capability document.

Appendix A:
(normative)
Time Attribute Requirements

Table 4 provides an overview of the requirements for use of the `at` and `from` attributes in ResourceSync documents. The top label in the column headings represents the `<sitemapindex>` root element for index documents, and the `<urlset>` root element for all other documents. The child label in the column headings represents the `<sitemap>` child element for index documents, and the `<url>` child element for all other documents.

The optional attributes `completed` and `until` are not shown in the table, but they may be added wherever the corresponding `at` and `from` attributes are mandatory, recommended, or optional.

Table 4 shows that, for example, a Change List must contain the `<rs:md>` child element of the `<urlset>` root element with the attribute `from` to convey the temporal interval covered by the Change List. The table also shows that the `<url>` child element of the `<urlset>` root element in a Change List must have the `<lastmod>` child element to convey the last modification time of a resource. Both mandatory attributes are, for example, shown in Example 19 in Section 12.1.

Table 4: Required and optional use of at and from attributes in ResourceSync documents

Capability Document	*/top*/rs:md/ @at	*/top*/rs:md/ @from	*/top/child/* rs:md/@at	*/top/child/* rs:md/@from	*/top/child/* lastmod
Resource List	Mandatory				Optional
Resource List Index	Mandatory		Optional		Optional
Resource Dump	Mandatory		Optional		Optional
Resource Dump Index	Mandatory		Optional		Optional
Resource Dump Manifest	Mandatory				Optional
Change List		Mandatory			Mandatory
Change List Index		Mandatory		Recommended	Optional
Change Dump		Mandatory		Recommended	Optional
Change Dump Index		Mandatory		Recommended	Optional
Change Dump Manifest		Mandatory			Mandatory

Bibliography

(This Bibliography is not part of the *ResourceSync Framework Specification*, ANSI/NISO Z39.99-2014. It is included for information only.)

[arXiv] arXiv.org [website]. Available at: http://arxiv.org/

[HTML Links] Raggett, Dave, Arnaud Le Hors, and Ian Jacobs, eds. "Links." Section 12 in: *HTML 4.01 Specification*. W3C Recommendation. World Wide Web Consortium, December 24, 1999. Available at: http://www.w3.org/TR/html401/struct/links.html

[JSON-Patch] Bryan, P., and M. Nottingham, eds. *JSON Patch*. Internet Draft. Internet Engineering Task Force (IETF), January 20, 2013. Available at: http://tools.ietf.org/html/draft-ietf-appsawg-json-patch-10

[OAI-PMH] Lagoze, Carl, Herbert Van de Sompel, Michael Nelson, and Simeon Warner, eds. *The Open Archives Initiative Protocol for Metadata Harvesting*. Version 2.0. Open Archives Initiative, December 7, 2008. Available at: http://www.openarchives.org/OAI/openarchivesprotocol.html

[ORE] Lagoze, Carl, Herbert Van de Sompel, Pete Johnston, Michael Nelson, Robert Sanderson, and Simeon Warner, eds. *ORE Specification - Abstract Data Model*. Open Archives Initiative, October 17, 2008. Available at: http://www.openarchives.org/ore/1.0/datamodel

[Relation Types] Klein, Martin, Robert Sanderson, Herbert Van de Sompel, Simeon Warner, Graham Klyne, Bernhard Haslhofer, Michael Nelson, and Carl Lagoze, eds. *Relation Types Used in the ResourceSync Framework*. Beta draft. Open Archives Initiative, September 27, 2013. Available at: http://www.openarchives.org/rs/relationtypes

[RFC 5261] Urpalainen, J. *An Extensible Markup Language (XML) Patch Operations Framework Utilizing XML Path Language (XPath) Selectors*. RFC 5261. Internet Engineering Task Force (IETF), September 2008. Available at: http://www.ietf.org/rfc/rfc5261.txt

[RFC 5785] Nottingham, M., and E. Hammer-Lahav. *Defining Well-Known Uniform Resource Identifiers (URIs)*. RFC 5785. Internet Engineering Task Force (IETF), April 2010. Available at: http://www.ietf.org/rfc/rfc5785.txt

[RFC 6596] Ohye, M., and J. Kupke. The Canonical Link Relation. RFC 6596. Internet Engineering Task Force (IETF), April 2012. Available at: http://www.ietf.org/rfc/rfc6596.txt

[rsync] *rsync*. From Wikipedia, the free encyclopedia. Available at: http://en.wikipedia.org/wiki/Rsync

[Web Architecture] Jacobs, Ian, and Norman Walsh, eds. *Architecture of the World Wide Web, Volume One*. W3C Recommendation. World Wide Web Consortium, December 15, 2004. Available at: http://www.w3.org/TR/webarch/

[XHTML Links] McCarron, Shane, et al., eds. "Link Module." Section 5.19 in: *XHTML™ Modularization 1.1*. Second Edition. World Wide Web Consortium, July 29, 2010. Available at: http://www.w3.org/TR/xhtml-modularization/abstract_modules.html#s_linkmodule