2013

A Technical Framework for Resource Synchronization

Martin Klein

Robert Sanderson

Herbert Van de Sompel

Simeon Warner

Bernhard Haslhofer

See next page for additional authors

Original Publication Citation

Klein, M., Sanderson, R., Van de Sompel, H., Warner, S., Haslhofer, B., Lagoze, C., & Nelson, M. L. (2013). A technical framework for resource synchronization. *D-Lib Magazine, 19*(1/2), 3 (1-15). doi:10.1045/january2013-klein

D-Lib Magazine

January/February 2013
Volume 19, Number 1/2
Table of Contents

A Technical Framework for Resource Synchronization

Martin Klein, Robert Sanderson, Herbert Van de Sompel
Los Alamos National Laboratory
{mklein, rsanderson, herbertv}@lanl.gov

Simeon Warner, Bernhard Haslhofer
Cornell University
{simeon.warner, bernhard.haslhofer}@cornell.edu

Carl Lagoze
University of Michigan
clagoze@umich.edu

Michael L. Nelson
Old Dominion University
mln@cs.odu.edu

doi:10.1045/january2013-klein

Printer-friendly Version

Abstract

This is the second paper in *D-Lib Magazine* about the ResourceSync effort conducted by the National Information Standards Organization (NISO) and the Open Archives Initiative (OAI). The first part provided a perspective on the resource synchronization problem and introduced a template that organized possible components of a resource synchronization framework in a modular manner. This paper details a technical framework devised using that template.

1 ResourceSync Capabilities

A previous paper published in *D-Lib Magazine* about the NISO/OAI ResourceSync effort [17] provided a perspective on the resource synchronization problem. The paper broke the problem down into several component problems, and suggested that each of those might be addressed by distinct technologies that could be stitched together in a modular manner to construct concrete Web-based synchronization frameworks that meet communities' requirements. The paper summarized the problem domain in a template that is repeated in Figure 1 but is now overlaid with the names of modular **capabilities** that systems may choose to implement in order to allow third parties to remain in sync with their evolving resources: Resource List, Resource Dump, Change List, Change Dump, Resource Dump Archive, Change List Archive, and Change Dump Archive.

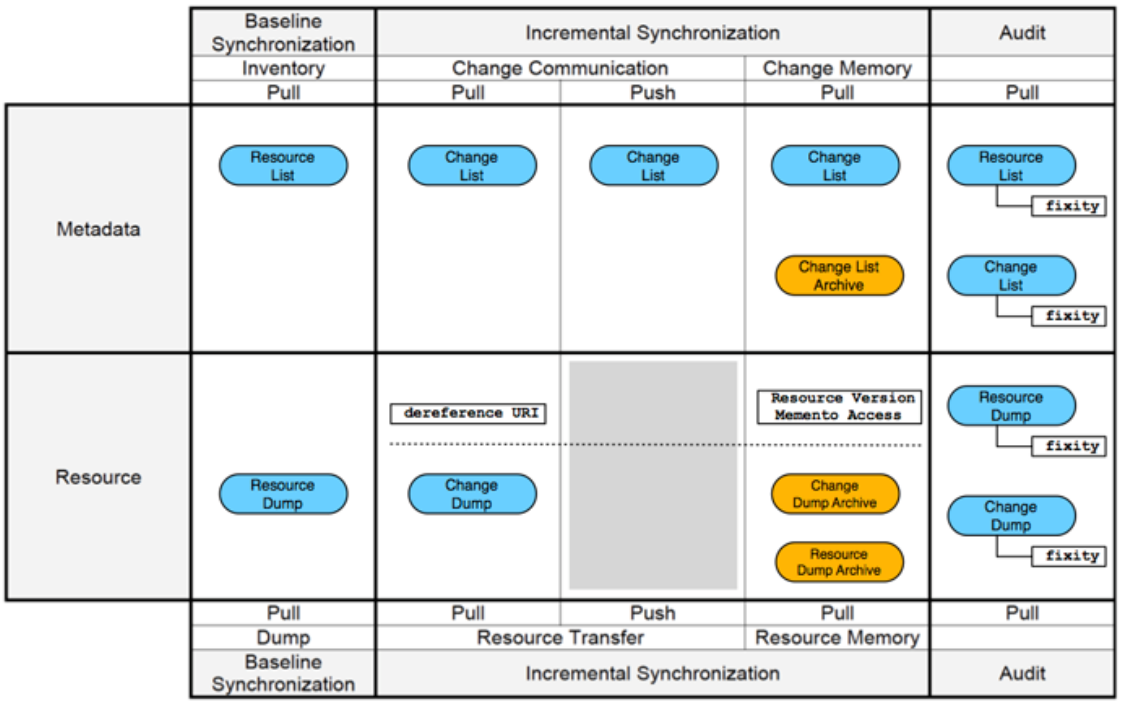

Figure 1: The ResourceSync template
(Larger View)

The following terms are introduced:

- **Source** — A server that hosts resources to be synchronized.
- **Destination** — A system that retrieves those resources to synchronize itself with the Source.

The resources to be synchronized require the possibility of associating both metadata and further resources with them. These requirements include:

- **URI** — Each resource must have a URI at which it is available.
- **Resource Type** — Expressing the media type of a resource is useful to support an understanding of the nature of resource representations.
- **Change Type** — Expressing the type of change a resource underwent (create, update, delete) is important to allow a Destination to understand the kind of action it needs to undertake to remain synchronized with the Source.
- **Modification Timestamp** — Recording the timestamp of a change to a resource is essential in order to allow a Destination to process changes in the appropriate temporal order.
- **Fixity Information** — This includes several possible pieces of information, including the size of the bitstream or a digest or hash such as an MD5 sum.
- **Links** — Links to mirror copies, alternative serializations, related services and canonical reference versions are all desirable functionality.

The core capabilities introduced by ResourceSync are:

- **Resource List** — A Source can recurrently publish an up-to-date Resource List in order to enumerate and describe its resources. Per resource, a Resource List contains its URI and optional metadata and links. A Destination uses a Resource List for synchronization by dereferencing the listed URIs.
- **Resource Dump** — In order to make its data available for bulk download, a Source can publish a Resource Dump. A Resource Dump is a document that points at packaged bitstreams associated with the resources hosted by the Source along with their metadata. A Resource Dump provides Destinations with an efficient method to obtain the Source's data with a limited amount of requests, sometimes only one, other times just a few.

- **Change List** — A Source can publish a Change List, which provides information about changes to the Source's resources, in order to decrease synchronization latency and reduce communication overhead. It is up to the Source to determine the temporal interval that is covered by a Change List. For example, a Change List could describe all changes of one day, one hour, or simply a fixed number of changes. A Change List conveys at least the URI of the changed resource, its last modification time, and the type of change (create, update, delete). A Change List can be made available under a pull paradigm, but can also be pushed to Destinations, for example, using a publish/subscribe approach. The latter allows Destinations to remain informed about a Source's changing data in near real-time.
- **Change Dump** — In order to make content changes available for download, a Source can publish a Change Dump. A Change Dump is a document that points at packaged bitstreams, whereby each bitstream is associated with a change that occurred to a Source's resource. The package also contains metadata about each change. In essence, a Change Dump is a Change List in which the content is provided by value instead of by reference.

The framework also includes capabilities that address memory requirements: **Change List Archives**, **Resource Dump Archives**, and **Change Dump Archives** support Destinations to access the state of resources as they were further back in time, for example, to allow Destinations to catch up with changes after having been offline or to gather all, not only the current, version of resources.

Further, if a Source publishes fixity information about its resources, it provides the opportunity for a Destination to check the completeness and accuracy of its copies of the Source's resources. In the ResourceSync framework this operation is referred to as Audit.

2 Temporal View and Discovery of Capabilities

Figure 2 below provides a temporal perspective on how a Source would implement the capabilities described above. The left half of the figure displays a time line and events at times t1 through t6. A Resource List is published three times, at t2, t4, and t6. At t2 the description of the Source's resources fits in a single Resource List. At t4 and t6, however, multiple Resource Lists are required to convey the resources that the Source makes available for synchronization. The union of the Lists published at t4 represents the state of the Source's data at t4, and, similarly the Lists published at t6 cover the data available for synchronization at t6.

The Source also publishes three Resource Dumps during the displayed time period, at t1, t3 and t5, respectively. Similar to a Resource List, a Resource Dump represents the Source's state at the time it was created. For example, Resource Dump1 represents the Source's state at time t1 and Resource Dump2 represents its state at t3. If the Source wishes to address memory requirements, it can provide access to both the current and previous Resource Dumps by implementing a Resource Dump Archive, represented as an orange bubble in Figure 2.

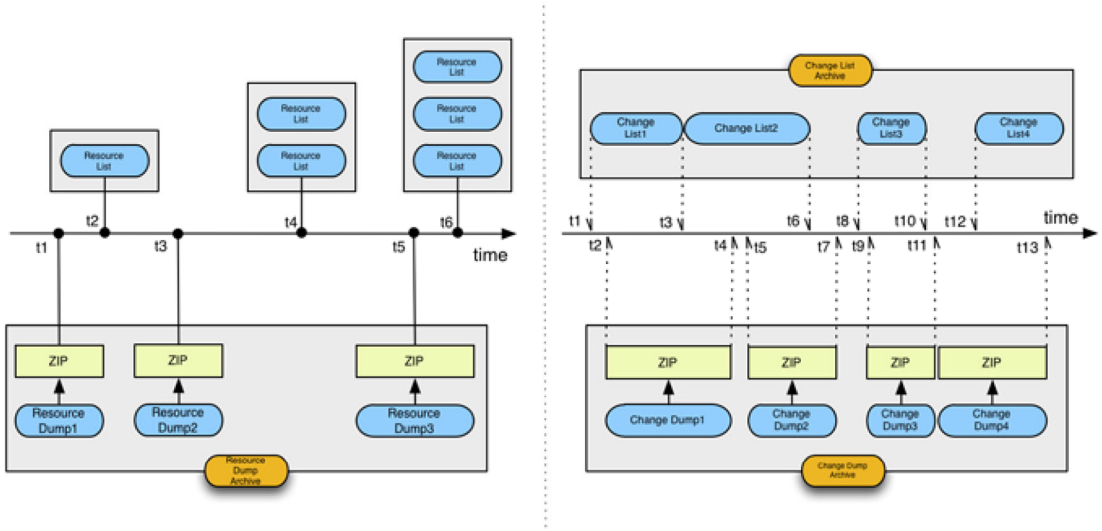

Figure 2: Temporal view of ResourceSync capabilities
(Larger View)

The right half of Figure 2 displays another time line with events pertaining to the communication of changes by a Source. Change List1, for example, covers all resource changes that occurred between times t1 and t3 at which point Change List2 continues to cover all changes until t6, and so forth. This perspective expresses the major temporal difference between a Resource List and a

Change List: A Resource List is a snapshot of the state of a Source's data taken at a discrete moment in time, whereas a Change List conveys the evolving state of the data over a given period of time. The Source can provide access to both the current and previous Change Lists by implementing a Change List Archive. This provides Destinations with the opportunity to process changes that they may have missed while being offline, for example.

The right part of Figure 2 also shows Change Dump1 referring to a ZIP file containing bitstreams associated with resource changes that occurred between t2 and t4, along with three other Change Dumps that cover changes that occurred in different temporal intervals. As can be seen, Change Dumps cover a period of time that can be disjoint from the time periods covered by Change Lists. For a Source to provide access to more than one Change Dump, it needs to implement a Change Dump Archive.

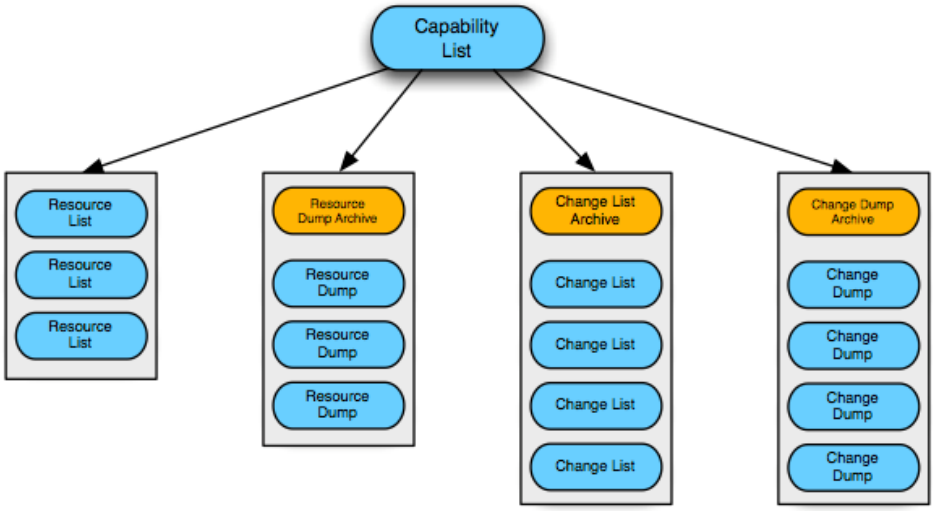

Figure 3: Discovery of and relations between ResourceSync capabilities

Figure 3 provides yet another perspective on ResourceSync capabilities, this time focused on conveying which of the modular capabilities a Source supports. To that end, the **Capability List** is introduced as a document that links (shown as arrows in Figure 3) to a Source's implementations of ResourceSync capabilities. The Capability List itself can be made discoverable in various ways, including a ResourceSync-specific well-known URI pattern [8], HTTP link headers [7], and (X)HTML links.

3 Resource Synchronization Use Cases

A general overview of the types of use cases that are considered in scope of the ResourceSync effort is provided in [4]. Here, in order to illustrate the framework and to emphasize its modular nature, a brief analysis of two typical, yet very different, synchronization use cases is offered: the pre-print repository arXiv.org and the linked data project DBpedia Live. The two cases are distinct in terms of resource volume, resource change frequency, and synchronization latency requirements and hence are likely to implement different capabilities of the ResourceSync framework.

The arXiv repository of scholarly articles has mirrors around the world. At the time of writing, the arXiv repository comprises about 2.4 million resources (articles with multiple versions and separate metadata records) and each year about 75,000 new articles and revisions to existing articles are added. Its current synchronization mechanism is based on a homegrown notification and audit system that replicates content to other trusted repositories. With the implementation of the ResourceSync framework, arXiv would not only be able to stay more closely synchronized with such trusted repositories but would also enable non-associated third parties to mirror its content.

The arXiv repository has manageable scale and fewer than 2,000 resource changes per day, which occur all at one time. Publishing a daily Resource List and periodic Resource Dump would be a reasonable base level of adoption of the ResourceSync framework. Destinations could use the Resource List for baseline synchronization by dereferencing all URIs it contains. By publishing a Resource Dump, arXiv could make the baseline synchronization process more efficient. To improve the efficiency of repeated synchronization, arXiv might publish a Change List or Change Dump shortly after the daily updates.

The DBpedia Live knowledge base is a repository of a different scale. It results from extracting information from Wikipedia and making it accessible to semantic web applications. Its English version alone contains descriptions of 3.77 million "things" (version 3.8) and, on average, undergoes two changes per second. Like arXiv, DBpedia Live offers homegrown synchronization functionality. It consists of publishing a monthly dump of the entire DBpedia Live database, and, at high but unpredictable frequency, publishing files that list changed and deleted RDF triples, respectively. Using the ResourceSync framework, the dump

capability could be maintained, yet be provided by means of a Resource Dump that is implemented and discoverable uniformly across information systems. A Resource List may not be an attractive option, mainly because the dereferencing of such a vast number of URIs would be prohibitive for both Source and Destinations. Change Dumps are rather similar to DBpedia Live's files with changed and deleted triples as they are a "by value" way to provide bitstreams for resources that changed during a given time period. In ResourceSync, such a bitstream can be a representation of an entire changed resource or it can be the difference between its previous and new representation. The latter is achieved by using a media type for the bitstream that identifies an approach to patch the previous bitstream of the resource with the provided diff bitstream to arrive at the new bitstream. Examples of such medias types are application/json-patch for patching JSON files [1] and application/patch-ops-error+xml for patching XML files [16]. DBpedia Live could actually use the former when representing RDF triples as JSON-LD [15]. This approach still adheres to a pull paradigm, yielding uncertainty as to when Destinations need to check for changes. In order to reduce latency, DBpedia Live could adopt Change Lists, provided under a publish/subscribe paradigm such as XMPP PubSub [5], allowing changes to be communicated as they occur.

4 Serialization Alternatives for Capabilities

All capabilities require documents to convey information pertaining to resources, and the choice of an appropriate document format that can be used throughout the framework has been the subject of explorations ever since the ResourceSync effort started. Three avenues were pursued:

1. Motivated by the similarity between the functionality intended by Resource Lists and Sitemaps, adopt and extend the Sitemap document formats (Sitemap and SitemapIndex) [11].
2. Motivated by the notion of continuous updating shared by Change Lists and feed technology, adopt and extend the Atom Syndication Format [9].
3. To avoid constraints anticipated by adopting and tailoring existing formats, introduce a ResourceSync-specific document format.

Sitemap-based approach

Sitemaps are widely used to advertise a server's resources to search engines and hence provide a functionality similar to the one intended by Resource Lists. Adoption of the Sitemap document formats would make many thousands of servers instantly compliant with one of the ResourceSync capabilities. Sitemaps are extensible by allowing child elements from foreign namespaces for the <*url*> element, which could be used by ResourceSync to, for example, convey the nature of a resource change, fixity information, links to related resources, etc. These features make a Sitemap-based approach appealing.

However, a number of issues caused concern. It needs to be recognized that the primary intent of a Sitemap is to advertise resources to search engines to support their discovery. The set of resources advertised to this end might be disjoint from the set a Source wishes to make available for synchronization. Also, technically, there are limits to the extensibility of Sitemaps; the Sitemap and SitemapIndex XML Schema ([12, 13]) do not allow foreign child elements of the root elements nor attributes on any of the elements used in Sitemaps and SitemapIndex. More importantly, experiments conducted with Google's webmaster tools, aimed at understanding how Google would consume Sitemaps with ResourceSync extensions revealed further problems:

- <*link*> and <*meta*> elements are expected to be in the XHTML namespace.
- <*meta*> elements have to have a "*content*" and a "*name*" attribute.
- Non HTTP URIs such as URNs are not accepted in <*loc*> elements.
- Google is very aggressive in detecting, following and indexing links in the Sitemap that are not provided by <*loc*> elements but by <*link*> elements.

The last point might be a major issue if a Source provides references whose targets (e.g. mirror copies) are not supposed to be indexed by search engines.

Examples for the Sitemap-based serialization can be seen in the following Section.

Atom-based approach

Motivated by the observation that the Atom Syndication Format [9] seems to be a good fit for the ongoing communication of resource changes as intended by Change Lists, an effort was made to explore the reuse of the format. Atom is widely used, and concepts introduced as Atom extensions [6], such as archived feeds, complete feeds and feed pagination are appealing. However, the fact that Atom is, in effect, the combination of a feed technology and a metadata format geared towards news syndication results in a significant cost without obvious benefits for a ResourceSync deployment:

- While the mandatory <*id*> and <*title*> elements at the feed level might be defensible, having both mandatory at the entry level is a burden for the ResourceSync framework. Using meaningful values, such as tag URIs [2] or urn:uuids [3] for the entry <*id*> might bring some value for Change Lists as they could uniquely identify change events and help Destinations to distinguish more easily between processed and unprocessed changes. However, such identifiers compress poorly adding further to the payload of the already verbose Atom serialization. Also, these identifiers have no obvious value for other capabilities, e.g. Resource Lists. Given their questionable value, Sources may decide to populate <*id*> elements with meaningless values, thereby negatively impacting Destination processes that assume they are meaningful and hence rely on them.
- There are two options for how to use Atom's content model to convey the URI of resources. Either an empty <*content*> element with a "*src*" attribute is included in the <*entry*> element, or a <*link*> element is used to convey the desired information. The first case requires inclusion of a mandatory <*summary*> element and the second option requires the use of a mandatory <*link*> element with the relation type "*alternate*". In combination with other links to related resources as required by ResourceSync use cases, the latter would lead to significant confusion in the interpretation of the information conveyed.
- The <*author*> element is mandatory for <*entry*> elements, yet expressing resource-level authorship has no obvious benefits in terms of resource synchronization. The requirement can be met by using a single <*author*> element at the feed level in which case the authorship is inherited by all entries. This is problematic because the author expressed at the feed level is likely not the author of the content described by an entry.

The result of the Atom explorations led even members of the ResourceSync Technical Committee with longstanding experience in the use of Atom to conclude that the format was inappropriate for the effort.

Examples of the Atom serialization can be found in the Appendix. Figure 9 displays a link-based Atom serialization and Figure 10 shows an Atom serialization based on <*content*> elements.

ResourceSync-specific approach

Given the drawbacks of Sitemaps and Atom, an effort was launched to investigate a document format tailored to the specific needs of ResourceSync. To that end, the information elements required by the various use cases detailed in [4] were identified and an XML-based syntax to express them was devised.

While this approach resulted in a compact, tailor-made document format, the obvious drawback is the predictable barrier of adoption. The introduction of a new XML format would yield an uphill adoption battle when compared to the reuse of widely deployed formats such as Sitemaps or Atom for which a community of practice and a choice of off-the-shelf tools and libraries already exist.

An example of the ResourceSync specific format can be seen in Figure 11 of the Appendix.

Decision

Despite the concerns raised by the prospect of reusing the Sitemap document formats, their adoption is more appealing than adopting Atom or devising a ResourceSync-specific format. The Technical Committee therefore decided to move forward with a Sitemap-based serialization approach and to ameliorate the identified concerns by:

- Liaising with the maintainers of the Sitemap protocol to work towards increased extensibility of the Sitemap and SitemapIndex formats.
- Avoiding unintended behavior by crawlers that consume Sitemaps by introducing a discovery approach that clearly distinguishes between regular Sitemap use and use of Sitemaps for the purpose of ResourceSync.

5 Serialization for ResourceSync Capabilities

The serialization used for the four main capabilities of the framework is described in the following sections.

Resource List

A Source can enable baseline synchronization by providing a list of its resources. In a Resource List, the URI of the resource is mandatory and additional information can be provided. Figure 4 below shows a simple Resource List with the description of two resources. It has the Sitemap specific <*urlset*> root element and one encapsulating <*url*> element for each described resource. The URI of a resource is provided in the <*loc*> element and its (optional) last modification date in the <*lastmod*> element.

The ResourceSync framework adds metadata for the document, and the listed resources:

- The top level <*rs:md*> element describes the document, including which capability it enables and its last modification date. This element is mandatory in all ResourceSync documents.
- The <*rs:md*> child element for each <*url*> element contains additional information about the described resource. The semantics of most of its attributes are inherited from [14] and [9].

The slightly cryptic element names <*rs:md*> and <*rs:ln*> have been chosen in preference to possibilities such as <*rs:meta*> and <*rs:link*> because our experiments showed that some services do namespace-blind parsing of Sitemap files and thus might mistake the elements for similarly named ones in other namespaces.

```
<?xml version="1.0" encoding="UTF-8"?>
<urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9"
        xmlns:rs="http://www.openarchives.org/rs/">
    <rs:md capability="resourcelist"
           modified="2012-10-31T09:00:00Z"/>
    <url>
        <loc>http://example.com/res1</loc>
        <lastmod>2012-10-30T14:00:00Z</lastmod>
        <rs:md hash="md5:1584abdf8ebdc9802ac0c6a7402c03b6"
               length="8876"
               etag="114c12b-213f-4ccb066bfd140"
               type="text/html"/>
    </url>
    <url>
        <loc>http://example.com/res2</loc>
        <lastmod>2012-10-30T13:00:00Z</lastmod>
        <rs:md hash="md5:1e0d5cb8ef6ba40c99b14c0237be735e
               sha-256:854f61290e2e197a11bc91063afce22e43f8ccc655237050ace766adc68dc784"
               length="14599"
               etag="114c070-21b5-4ca7e085f5c80"
               type="application/pdf"/>
    </url>
</urlset>
```

Figure 4: A Resource List

The Sitemap specification has a limit of 50,000 URLs per Sitemap, but this can be increased to 2.5 billion by nesting Sitemaps using a SitemapIndex. The ResourceSync framework adopts this approach directly.

Resource Dump

A Resource Dump description is implemented like a Sitemap with the <*urlset*> root element, and each child <*url*> element pointing to a ZIP file that packages bitstreams and a Resource Dump Manifest that describes them.

The Manifest is very similar to the Resource List, with the URIs of the resources being conveyed in the <*loc*> element along with additional metadata but must also include a "*path*" attribute for each resource that provides a pointer to the file path of the bitstream within the ZIP file. The manifest file must be named "*manifest.xml*" and be located in the root of the ZIP file's directory structure. By nesting Manifests, up to 2.5 billion bitstreams can be provided in a ZIP file.

```
<?xml version="1.0" encoding="UTF-8"?>
<urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9"
        xmlns:rs="http://www.openarchives.org/rs/">
    <rs:md capability="resourcedump-manifest"
           modified="2012-10-31T19:00:00Z"/>
    <url>
        <loc>http://example.com/res1</loc>
        <lastmod>2012-10-30T14:00:00Z</lastmod>
        <rs:md hash="md5:1584abdf8ebdc9802ac0c6a7402c03b6"
               path="/resources/res1"/>
```

```
    </url>
    <url>
        <loc>http://example.com/res2</loc>
        <lastmod>2012-10-30T13:00:00Z</lastmod>
        <rs:md hash="md5:1e0d5cb8ef6ba40c99b14c0237be735e"
               path="/resources/res2"/>
    </url>
</urlset>
```

Figure 5: A Resource Dump Manifest

The Source may also maintain a list of previous Resource Dumps as a Resource Dump Archive. This is implemented as a SitemapIndex, as shown previously in Figures 1, 2, and 3.

Change List

A Change List enables a Source to communicate changes to individual resources (one communication per create, update, delete event) rather than providing a snapshot in time of all its resources. A Destination can obtain and process Change Lists from the Source in order to determine which resources it needs, if any, to update its own repository. A Source can push the Change List to Destinations via publish/subscribe technology, further reducing the latency before the Destination is synchronized.

```
<?xml version="1.0" encoding="UTF-8"?>
<urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9"
        xmlns:rs="http://www.openarchives.org/rs/">
    <rs:md capability="changelist"
           modified="2012-10-31T11:00:00Z"/>
    <url>
        <loc>http://example.com/res1</loc>
        <lastmod>2012-10-30T18:00:00Z</lastmod>
        <rs:md hash="md5:1584abdf8ebdc9802ac0c6a7402c03b6"
               change="updated"/>
    </url>
    <url>
        <loc>http://example.com/res2</loc>
        <lastmod>2012-10-29T13:00:00Z</lastmod>
        <rs:md hash="md5:1e0d5cb8ef6ba40c99b14c0237be735e"
               change="created"/>
    </url>
    <url>
        <loc>http://example.com/res3.tiff</loc>
        <lastmod>2012-10-29T11:00:00Z</lastmod>
        <rs:md change="deleted"/>
    </url>
    <url>
        <loc>http://example.com/res4.png</loc>
        <lastmod>2012-10-29T03:00:00Z</lastmod>
        <rs:md hash="sha-256:f4OxZX_x_DFGFDgghgdfb6rtSx-iosjf6735432nklj"
               type="image/png"
               change="updated"/>
        <rs:ln rel="duplicate"
               href="http://mirror1.example.com/res4.png"
               modified="2012-10-29T03:00:00Z"/>
        <rs:ln rel="alternate"
               href="http://example.com/res4.jpg"
               modified="2012-10-29T03:00:00Z"
               type="image/jpeg"/>
    </url>
</urlset>
```

Figure 6: A Change List

Figure 6 shows an example of a Change List, implemented with the <*urlset*> root element. Each changed resource is described within a <*url*> element, its URI is provided with the <*loc*> element and the timestamp for the change is conveyed by the <*lastmod*> element. Both values, as well as the change type, are mandatory in Change Lists. The type of change is indicated with the "*change*" attribute in the <*rs:md*> element. Further information about the changed resource can be provided, as in previous examples.

For cases where a Source wishes to convey additional information about a resource such as its preferred download location or an alternate representation, it can use the <*rs:ln*> element as a child of the <*url*> element. The provided references should have the appropriate relation type such as "*duplicate*" or "*alternate*", as seen in Figure 6. This child element can also be used if a Source wishes to only convey information about the part of the resource that has actually changed.

For a Source to hold on to more than 50,000 URLs in its Change List, it needs to implement a Change List Archive in the form of a SitemapIndex. This Archive can then refer to up to 50,000 Change Lists. Using this technique, a Source that consistently encounters 14 resource changes per second and which completely fills each Change List, can provide descriptions for over 5 years before having to drop the first hour's worth of changes.

Change Dump

Similar to the Resource List/Resource Dump resemblance, a Source can provide the representations of the changed resources with a Change Dump. A Change Dump is implemented in the form of a Sitemap that points to one or more ZIP files that contain bitstreams associated with changed resources, and a Manifest included in the ZIP as per the Resource Dump capability, and displayed in Figure 7. A Change Dump provides the Destination with an opportunity to obtain changed resources more efficiently than using Change Lists as it requires significantly fewer HTTP requests.

```
<?xml version="1.0" encoding="UTF-8"?>
<urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9"
        xmlns:rs="http://www.openarchives.org/rs/">
    <rs:md capability="changedump-manifest"
           modified="2012-10-30T21:00:00Z"/>
    <url>
        <loc>http://example.com/res1.html</loc>
        <lastmod>2012-10-30T14:00:00Z</lastmod>
        <rs:md hash="md5:0988647082c8bc51778894a48ec3b576"
               length="5426"
               etag="80102-20b-4b4247135838c"
               type="text/html"
               change="updated"
               path="/changes/res1.html"/>
    </url>
    <url>
        <loc>http://example.com/res2.pdf</loc>
        <lastmod>2012-10-30T11:00:00Z</lastmod>
        <rs:md hash="md5:f906610c3d4aa745cb2b986f25b37c5a
               sha-256:f138185cddef488264a0323aee56e7647e89cd7a4d6e45ba28b3be26234a6d09"
               length="38297"
               etag="415706-1b2a-4be234cb66fc0"
               type="application/pdf"
               change="updated"
               path="/changes/res2.pdf"/>
    </url>
    <url>
        <loc>http://example.com/res3.tiff</loc>
        <lastmod>2012-10-30T09:00:00Z</lastmod>
        <rs:md change="deleted"/>
    </url>
    <url>
        <loc>http://example.com/res1.html</loc>
        <lastmod>2012-10-28T11:00:00Z</lastmod>
        <rs:md hash="md5:1c1b0e264fa9b7e1e9aa6f9db8d6362b"
               length="4339"
               etag="92304-29g-5y3329684638c"
```

```
                type="text/html"
                change="created"
                path="/changes/res1.html"/>
    </url>
</urlset>
```

Figure 7: A Change Dump Manifest

A Source can also implement a Change Dump Archive as a SitemapIndex if it wishes to hold on to and link to Change Dumps that cover previous time periods.

Capability List

In order to make use of these capabilities, a Destination needs to discover the URIs for the describing documents, which it does via a Capability List, as shown in Figure 3 above. The Capability List also follows the Sitemap syntax with the <*urlset*> root element and each capability having a <*url*> element.

Figure 8 shows an example of a Capability List. It includes pointers to the Resource List, Resource Dump, Change List, and Change Dump documents.

```
<?xml version="1.0" encoding="UTF-8"?>
<urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9"
        xmlns:rs="http://www.openarchives.org/rs/">
    <rs:md capability="capabilitylist"
           modified="2012-10-30T14:00:00Z"/>
    <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>
```

Figure 8: A Capability List document

Each Capability List describes the capabilities of a single collection of resources, however, a Source may have multiple collections. The grouping of collections is achieved by using a Capability List Index, again in form of a SitemapIndex. This way a Source can maintain up to 50,000 collections and refer to their corresponding Capability Lists.

Linking between capabilities

The Sitemap protocol does not provide the means for a Destination to navigate from a Sitemap-structured document with the <*urlset*> root element to its parent document with the <*sitemapindex*> root element. ResourceSync introduces this possibility by using a child element <*rs:ln*> of <*urlset*> that expresses an "*up*" relation, enabling clients to easily traverse the tree of documents. A second link, with the relation "*top*", is also included in each document to reference the Capability List, such that clients can discover the other supported capabilities.

6 Summary

This article has provided a technical overview of the emerging ResourceSync framework. It has introduced the synchronization capabilities of the framework by putting them in the perspective of a template that breaks the resource synchronization problem down into component problems. It also provided a temporal and a discovery perspective of the framework. The article further described the explorations that led to adopting and enhancing the Sitemap document formats as the basis of ResourceSync and it provided examples of documents used for four core capabilities.

The evolving ResourceSync specification is available at [10]. A beta version will be released in the course of January 2013. Broad public feedback to that version will be solicited and test implementations for it are planned. The insights gained from both will be rolled into a final version that should be available by Summer 2013.

7 References

[1] P. Bryan, M. Nottingham. JSON Patch. Draft, December 2012.

[2] T. Kindberg, S.Hawke. The 'tag' URI Scheme. RFC 4151, October 2005.

[3] P. Leach, M. Mealling, R. Salz. A Universally Unique IDentifier (UUID) URN Namespace. RFC 4122, July 2005.

[4] S. Lewis, R. Jones, S. Warner. Motivations for the Development of a Web Resource Synchronisation Framework. *Ariadne*, Issue 70, November 2012.

[5] P. Millard, P. Saint-Andre, R. Meijer. XEP-0060: Publish-Subscribe. July 2010.

[6] M. Nottingham. Feed Paging and Archiving. RFC 5005, September 2007.

[7] M. Nottingham. Web Linking. RFC 5988, October 2010.

[8] M. Nottingham, M. Hammer-Lahav. Defining Well-Known Uniform Resource Identifiers (URIs). RFC 5785, April 2010.

[9] M. Nottingham, R. Sayre. The Atom Syndication Format. RFC 4287, December 2005.

[10] ResourceSync Site

[11] Sitemaps XML Format.

[12] Sitemap Schema.

[13] SitemapIndex Schema.

[14] J. Snell. Atom Link Extensions. Draft 09, June 2012.

[15] M. Sporny, G. Kellogg, M. Lanthaler. JSON-LD Syntax. Draft, December 2012.

[16] J. Urpalainen. An Extensible Markup Language (XML) Patch Operations Framework Utilizing XML Path Language (XPath) Selectors. RFC 5261, September 2008.

[17] H. Van de Sompel, R. Sanderson, M. Klein, M. L. Nelson, B. Haslhofer, S. Warner, C. Lagoze. A Perspective on Resource Synchronization. *D-Lib Magazine*, Vol. 18, No. 9/10, 2012. http://dx.doi.org/10.1045/september2012-vandesompel.

8 Appendix

```
<?xml version="1.0" encoding="UTF-8"?>
<feed rs:type="http://www.openarchives.org/rs/changelist"
      xmlns="http://www.w3.org/2005/Atom"
      xmlns:rs="http://www.openarchives.org/rs/">
    <title>example.com Change List</title>
    <id>urn:uuid:609db020-3ccf-11e2-a25f-0800200c9a66</id>
    <link rel="self"
          href="http://example.com/mychangelist.xml"/>
    <link rel="http://www.openarchives.org/rs/capabilitylist"
          href="http://example.com/mycapabilitylist.xml"/>
    <updated>2012-10-31T09:00:00Z</updated>
    <author>
```

```
        <name>example.com resourcesync admin</name>
    </author>
    <entry>
        <title>resource 1</title>
        <id>urn:uuid:a422b110-3ccf-11e2-a25f-0800200c9a66</id>
        <updated>2012-10-30T18:00:00Z</updated>
        <link rel="canonical alternate"
              href="http://example.com/res1"
              hash="md5:1584abdf8ebdc9802ac0c6a7402c03b6"
              rs:change="updated"/>
    </entry>
    <entry>
        <title>resource 2</title>
        <id>urn:uuid:a422b111-3ccf-11e2-a25f-0800200c9a22</id>
        <updated>2012-10-29T13:00:00Z</updated>
        <link rel="canonical alternate"
              href="http://example.com/res2 "
              hash="md5:1e0d5cb8ef6ba40c99b14c0237be735e"
              rs:change="created"/>
    </entry>
    <entry>
        <title>resource 3</title>
        <id>urn:uuid:91730420-3cd9-11e2-a25f-0800200c9a66</id>
        <updated>2012-10-29T11:00:00Z</updated>
        <link rel="canonical alternate"
              href="http://example.com/res3.tiff"
              rs:change="deleted"/>
    </entry>
    <entry>
        <title>resource 4</title>
        <id>urn:uuid:91730420-3cd9-11e2-a25f-0800200c9444</id>
        <updated>2012-10-29T03:00:00Z</updated>
        <link rel="canonical alternate"
              href="http://example.com/res4.png"
              hash="sha-256:f4OxZX_x_DFGFDgghgdfb6rtSx-iosjf6735432nklj"
              type="image/png"
              rs:change="updated"/>
        <link rel="duplicate"
              href="http://mirror1.example.com/res4.png"
              modified="2012-10-29T03:00:00Z"/>
        <link rel="alternate"
              href="http://example.com/res4.jpg"
              modified="2012-10-29T03:00:00Z"
              type="image/jpeg"/>
    </entry>
</feed>
```

Figure 9: Link-based Atom syntax considered for ResourceSync

```
<?xml version="1.0" encoding="UTF-8"?>
<feed rs:type="http://www.openarchives.org/rs/changelist"
      xmlns="http://www.w3.org/2005/Atom"
      xmlns:rs="http://www.openarchives.org/rs/">
    <title>example.com Change List</title>
    <id>urn:uuid:609db020-3ccf-11e2-a25f-0800200c9a66</id>
    <link rel="self"
          href="http://example.com/mychangelist.xml"/>
    <link rel="http://www.openarchives.org/rs/capabilitylist"
          href="http://example.com/mycapabilitylist.xml"/>
    <updated>2012-10-31T09:00:00Z</updated>
    <author>
```

```
        <name>example.com resourcesync admin</name>
    </author>
    <entry>
        <title>resource 1</title>
        <id>urn:uuid:a422b110-3ccf-11e2-a25f-0800200c9a66</id>
        <updated>2012-10-30T18:00:00Z</updated>
        <content src="http://example.com/res1"
                 rs:hash="md5:1584abdf8ebdc9802ac0c6a7402c03b6"
                 rs:change="updated"/>
        <summary>info about res1</summary>
    </entry>
    <entry>
        <title>resource 2</title>
        <id>urn:uuid:a422b111-3ccf-11e2-a25f-0800200c9a22</id>
        <updated>2012-10-29T13:00:00Z</updated>
        <content src="http://example.com/res2 "
                 rs:hash="md5:1e0d5cb8ef6ba40c99b14c0237be735e"
                 rs:change="created"/>
        <summary>info about res2</summary>
    </entry>
    <entry>
        <title>resource 3</title>
        <id>urn:uuid:91730420-3cd9-11e2-a25f-0800200c9a66</id>
        <updated>2012-10-29T11:00:00Z</updated>
        <content src="http://example.com/res3.tiff"
                 rs:change="deleted"/>
        <summary>info about res3</summary>
    </entry>
    <entry>
        <title>resource 4</title>
        <id>urn:uuid:91730420-3cd9-11e2-a25f-0800200c9444</id>
        <updated>2012-10-29T03:00:00Z</updated>
        <conent src="http://example.com/res4.png"
                rs:hash="sha-256:f4OxZX_x_DFGFDgghgdfb6rtSx-iosjf6735432nklj"
                rs:type="image/png"
                rs:change="updated"/>
        <summary>info about res4</summary>
        <link rel="duplicate"
              href="http://mirror1.example.com/res4.png"
              modified="2012-10-29T03:00:00Z"/>
        <link rel="alternate"
              href="http://example.com/res4.jpg"
              modified="2012-10-29T03:00:00Z"
              type="image/jpeg"/>
    </entry>
</feed>
```

Figure 10: Content-based Atom syntax considered for ResourceSync

```
<?xml version="1.0" encoding="UTF-8"?>
<resync xmlns="http://www.openarchives.org/rs/"
        rstype="http://www.openarchives.org/rs/changelist"
        modified="2012-10-31T09:00:00Z">
    <link rel=http://www.openarchives.org/rs/capabilitylist
          href="http://example.com/dataset/capabilitylist-index.xml"/>
    <item change="updated"
          href=http://example.com/res1
          hash="md5:1584abdf8ebdc9802ac0c6a7402c03b6"
          modified="2012-10-30T18:00:00Z"/>
    <item change="created"
          href=http://example.com/res2
```

```
            hash="md5:1e0d5cb8ef6ba40c99b14c0237be735e"
            modified="2012-10-29T13:00:00Z"/>
      <item change="deleted"
            href=http://example.com/res3.tiff
            modified="2012-10-29T11:00:00Z"/>
      <item change="updated"
            href=http://example.com/res4.png
            hash="sha-256:f4OxZX_x_DFGFDgghgdfb6rtSx-iosjf6735432nklj"
            modified="2012-10-29T03:00:00Z"
            type="image/png"/>
          <link rel="duplicate"
                href="http://mirror1.example.com/res4.png"
                modified="2012-10-29T03:00:00Z"/>
          <link rel="alternate"
                href="http://example.com/res4.jpg"
                modified="2012-10-29T03:00:00Z"
                type="image/jpeg"/>
</resync>
```

Figure 11: ResourceSync-specific syntax considered for ResourceSync

About the Authors

Martin Klein received his Diploma in Computer Science from the University of Applied Sciences Berlin (2002) and his Ph.D. in Computer Science from Old Dominion University (2011). From 2002 to 2005, he was a scientist at the University of Applied Sciences in Berlin conducting research in the realm of e-Learning and mobile computing. At Old Dominion University, he was part of the Web Science and Digital Libraries Research Group led by Dr. Michael L. Nelson, and a part-time lecturer in the Computer Science Department. He currently is a Postdoctoral Research Associate at the Research Library of the Los Alamos National Laboratory. His research interests include digital preservation, the temporal aspect of web resources, and information retrieval and extraction.

Robert Sanderson is an information scientist in the Research Library at Los Alamos National Laboratory and previously a Lecturer in Computer Science at the University of Liverpool. His areas of research are focused around scholarly communication, digital humanities and large scale data mining. He has won international awards for his research, including the 2010 Digital Preservation Award and both the Vannevar Bush Best Paper award at JCDL2011 and Best Poster Award at JCDL2012. Between 2009 and 2011, he was the UIUC GSLIS Honorary Research Fellow for his interdisciplinary work in digital humanities. He is an editor of several international specifications including, most recently, W3C Open Annotation Community Draft, IETF Memento Internet Draft, and the NISO Resource Synchronization specification.

Herbert Van de Sompel is an information scientist at the Los Alamos National Laboratory and for 10 years has led the Digital Library Research & Prototyping Team. The Team does research regarding various aspects of scholarly communication in the digital age, including information infrastructure, interoperability, digital preservation and alt-metrics. Dr. Van de Sompel has played a major role in creating the Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH), the Open Archives Initiative Object Reuse & Exchange specification (OAI-ORE), the OpenURL Framework for Context-Sensitive Services, the SFX linking server, the bX scholarly recommender service, and info URI. Currently, he works with his team on the Open Annotation, Memento and ResourceSync projects. In recognition of his contributions, he has received numerous awards including the SPARC Innovator Award and the Frederick G. Kilgour Award for Research in Library and Information Technology. He graduated in Mathematics and Computer Science at Ghent University, Belgium, and holds a Ph.D. in Communication Science from the same university. Herbert's web site is at http://public.lanl.gov/herbertv/home/.

Simeon Warner is Director of the Repositories Group at Cornell University Library. Current projects include development of an archival repository, the arXiv e-print archive (http://arxiv.org/), and Project Euclid (http://projecteuclid.org/). He was one of the developers of arXiv and his research interests include web information systems, interoperability, plagiarism detection, and open-access scholarly publishing. He has been actively involved with the Open Archives Initiative (OAI) since its inception and was one of the authors of the OAI-PMH and OAI-ORE specifications.

Bernhard Haslhofer is an EU Marie Curie Fellow Postdoc at Cornell University Information Science. His research interest lies in the area of global, decentralized data networks and their social, cultural, and technical contexts. Dr. Haslhofer designs, builds, and experiments with Web-based data infrastructures, works on solutions for assessing and maintaining data quality in open environments, and examines how we can (re-)use open data in application use cases. In his current research, he investigates how data networks and algorithms built on the principles of openness and decentralization could support scholarship and, vice versa, how knowledge produced in the scholarship cycle could flow into broader, possibly public data and knowledge networks. He earned a Ph.D. in Computer Science from University of Vienna.

Carl Lagoze is an associate professor at the School of Information at the University of Michigan. Prior to this appointment, he spent a large part of his academic and research career at Cornell University, lastly in the information Science department. His research focuses on data and scholarly cyberinfrastructure, investigating both the technical aspects and the social/cultural/policy influences on cyberinfrastructure. He has been a longtime collaborator on Open Archives Initiative projects including OAI-PMH, ORE, and the present resourceSync project.

Michael L. Nelson is an associate professor of computer science at Old Dominion University. Prior to joining ODU, he worked at NASA Langley Research Center from 1991-2002. He is a co-editor of the OAI-PMH, OAI-ORE, Memento, and ResourceSync specifications. His research interests include repository-object interaction and alternative approaches to digital preservation. More information about Dr. Nelson can be found at: http://www.cs.odu.edu/~mln/.