



Published on *Ariadne* (<http://www.ariadne.ac.uk>)

Introducing UnAPI

30 July 2006 - 12:00am

Table of Contents [\[hide\]](#)

1. [Barriers to Web Clipboard Integration](#)
2. [Initial Approach](#)
3. [The unAPI Specification](#)
4. [unAPI in Action: Examples](#)
5. [Sample record view:](#)
6. [Highlights of Issues Considered](#)
7. [Next Steps](#)
8. [References](#)
9. [Author Details](#)

Common Web tools and techniques cannot easily manipulate library resources. While photo sharing, link logging, and Web logging sites make it easy to use and reuse content, barriers still exist that limit the reuse of library resources within new Web services. [1][2] To support the reuse of library information in Web 2.0-style services, we need to allow many types of applications to connect with our information resources more easily. One such connection is a universal method to copy any resource of interest. Because the copy-and-paste paradigm resonates with both users and Web developers, it makes sense that users should be able to copy items they see online and paste them into desktop applications or other Web applications. Recent developments proposed in weblogs [3][4] and discussed at technical conferences [5] [6] suggest exactly this: extending the 'clipboard' copy-and-paste paradigm onto the Web. To fit this new, extended paradigm, we need to provide a uniform, simple method for copying rich digital objects out of any Web application.

Barriers to Web Clipboard Integration

The initial Microsoft Live Clipboard specification provides a straightforward way of accomplishing Web clipboard copy and paste. It uses a combination of JavaScript-based in-browser code and an XML wrapper for item content, to provide users with clipboard functionality for some common types of objects defined by microformat specifications [7]. Early

work using the Live Clipboard technique in the National Science Digital Library, a U.S. National Science Foundation programme, integrates this technique with more robust digital library protocols, providing for clipboard copy and paste of complex digital objects in software environments with robust architectures [8]. These are tremendously exciting innovations and demonstrate the potential of this approach.

The Live Clipboard demonstrations show copying of event or business card-like information by copying these simple objects between commonly used Web sites and desktop applications. The NSDL demonstration moves complex objects [9][10] between what are presumed to be scholarly communications tools. Both sets of demonstrations are alike in that they are driven by interface events whereby users click and choose to perform clipboard actions in menus. However, the Web 2.0 approach calls for an additional blurring of the lines between user- and machine-driven operations, so it is necessary to devise an approach that also allows software to drive clipboard-style copy and paste functions on users' behalf.

Requirements for Automating Object 'Copy'

In an automated processing model that supports scripted copying of objects found on Web pages, the following three functional criteria must be met:

- A standard way to identify individual objects on Web pages;
- A standard way to discover a path to an API for retrieving objects;
- A standard API to retrieve object copies in all available formats.

Without a standard way for software to identify objects on Web pages, scripts must resort to screen scraping and other unsustainable techniques for guessing where objects start and end. The same logic applies to the requirements of a standard way to find an API entry point, and a common definition of an API for retrieving objects. Without these, third-party applications have to hard-code or guess at the locations and protocols offered by the plethora of Web 2.0 and digital library APIs and their various implementations across the Web.

Almost There Already

Fortunately, the digital library community already comes close to satisfying each of these requirements. Protocols such as OAI-PMH [11] and OpenURL [12] each provide frameworks for implementing services that support standardised object fetching through an API. The COinS convention for embedding OpenURL ContextObjects in the HTML SPAN element [13] provides a standard way of identifying objects on Web pages whenever a ContextObject contains an identifier reference.

Initial Approach

Members of the gcs-pcs-list [14] first experimented in this area by starting with these digital library tools, because they were already available and well-known. To provide scriptable object copying from Web pages, we combined COinS (with identifiers) on Web pages with JavaScript-based calls to the OAI-PMH functions ListMetadataFormats and GetRecord. To enable

Javascript code to find OAI-PMH services automatically, we added HTML LINK tags pointing to relevant OAI-PMH services for our resources, following the pattern for feed auto-discovery now widely implemented across the Internet [15]. These experiments took the form of Greasemonkey scripts [16] which, upon finding COinS with identifiers and OAI-PMH LINK elements, would automatically query the OAI-PMH services' ListMetadataFormats functions and present users with direct links to OAI-PMH GetRecord functions for each available format [17].

This worked quite well, and provided an interesting set of demonstrations. We wrote connectors for a variety of well-known Web sites - the Library of Congress American Memory collections, the arXiv.org pre-print service, Google Books, Amazon.com, and more [18]. These connectors, combined with the 'get this item in formats X, Y, or Z' links that were automatically written into Web pages for users to click, showed great promise and interested our colleagues.

Problems Selling COinS and OAI-PMH

The main problem with this approach was the difficulty explaining how it worked, especially to those less familiar with library-specific technologies. Despite the widespread adoption of OpenURL and the proliferation of OAI-PMH-based content and service providers, few people in the library profession understand the steps necessary to implement these services, and far fewer people outside the library profession can successfully wade through the jargon necessary to understand either. Even if each were readily understood, still more barriers make this approach unlikely to succeed. Firstly, relatively few resources are actually available over OAI-PMH; among the few collections with OAI-PMH interfaces, most typically provide access only to metadata, not full objects [19]. Secondly, many OAI-PMH providers use item identifiers unique to metadata records, and therefore items are not cross-referenced by more widely-known content identifiers. Given these conditions, it would be a mistake to presume that this approach could quickly scale to provide bare object access to a much larger swathe of library resources. Ultimately we want to provide automated access to our resources through clearly defined and familiar techniques that can be implemented in only a few hours of work by a typical Web developer. These rapid implementations need to accommodate both the data provider making resources accessible as well as the downstream clients that need to access such resources. For such a framework to succeed, these techniques must be understandable to the Web community at large without prior knowledge of specific digital library standards.

The unAPI Specification

We addressed this problem by writing a much simpler specification that meets the requirements listed above and remains easy to understand and implement. The 'unAPI' specification [20] is less than two pages long and defines only three components, one to address each of the functional criteria listed previously:

1. An HTML convention for identifying objects in Web pages;
2. An HTML LINK tag for autodiscovery of an unAPI service relevant to items on a given page;
3. An API consisting of three HTTP functions to retrieve objects by their identifiers in any available format.

We developed this specification on the public gcs-pcs-list between January and June 2006. Revisions were released nearly every month during this period and made publicly available at <http://unapi.info> . At every revision stage, after discussing and deciding on issues collected along the way, participants developed at least three independent test implementations similar to those described in the next section. This helped ensure that the specification was indeed manageable, and it enabled us to understand the issues raised during implementation. We completed and published unAPI Version 1 on 23 June, 2006 [21].

unAPI in Action: Examples

The unAPI specification itself contains a simple informative example, excerpted here. The unAPI HTML convention for identifying objects in Web pages is patterned after the technique developed by the microformats.org community for combining machine-readable data values as attribute values in ABBR elements with human-readable representations of that data as text content inside the ABBR:

```
<abbr class="unapi-id"
      title="http://unapi.info/news/archives/9"></abbr>
```

The unAPI LINK autodiscovery pattern mimics the pattern used by Web browsers to discover news feeds:

```
<link rel="unapi-server"
      type="application/xml" title="unAPI"
      href="http://unapi.info/news/unapi.php" />
```

The unAPI HTTP functions comprise a 'list all object formats' function with no parameters, a 'list formats for a particular object' function with an identifier parameter, and a 'get a particular format for a particular object' function with identifier and format parameters. The first two functions return a simple XML response listing formats that are supported for all items available from the unAPI service. For example, a call to an unAPI service such as this:

```
http://unapi.info/news/unapi.php?id=http://unapi.info/news/archives/9
```

...might return an XML response like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<formats id="http://unapi.info/news/archives/9">
<format name="oai_dc" type="application/xml"
docs="http://www.openarchives.org/OAI/2.0/oai_dc.xsd"/>
<format name="mods" type="application/xml" />
</formats>
```

The following examples all implement unAPI Version 1. Each example includes a link to user-visible records and unAPI links to one of the records in that view. Follow the links to see for yourself what unAPI looks like.

unAPI in OPACs and Other Databases

- Evergreen: The main OPAC for the Evergreen ILS (Integrated Library System) inserts unAPI identifiers into all result and detail pages, allowing users of unAPI harvesting tools to extract and reuse records from our catalogue in new and interesting ways. In addition to the main OPAC, the text-only alternate OPAC for Evergreen uses OpenSearch [22] and unAPI exclusively for the display of search and detail results. The HTML version of the hit list from OpenSearch is generated as an Atom XML [23] feed with identifiers suitable for unAPI resolution, and an XSLT transformation is applied to create unAPI links to the record and holdings page. The unAPI autodiscovery link is inserted into the HTML head, and unAPI identifiers, supplied inside the Atom entity elements, are used to build direct links to the bibliographic and holdings data. By leveraging unAPI, OpenSearch and Atom in this way, we provide a mechanism for automated harvesting of the exposed data as well as enabling end-user tools for building local bibliographies.

Sample record view:

<http://dev.gapines.org/opac/extras/opensearch/1.1/-/html-full/title/Tales+of+the+gross+and+gruesome>

Sample unAPI formats list:

[http://dev.gapines.org/opac/extras/unapi?id=tag:dev.gapines.org,2006:biblio-record_entry/307171/-](http://dev.gapines.org/opac/extras/unapi?id=tag:dev.gapines.org,2006:biblio-record_entry/307171/)

Sample object via unAPI:

http://dev.gapines.org/opac/extras/unapi?id=tag:dev.gapines.org,2006:biblio-record_entry/307171/-&format=marcxml

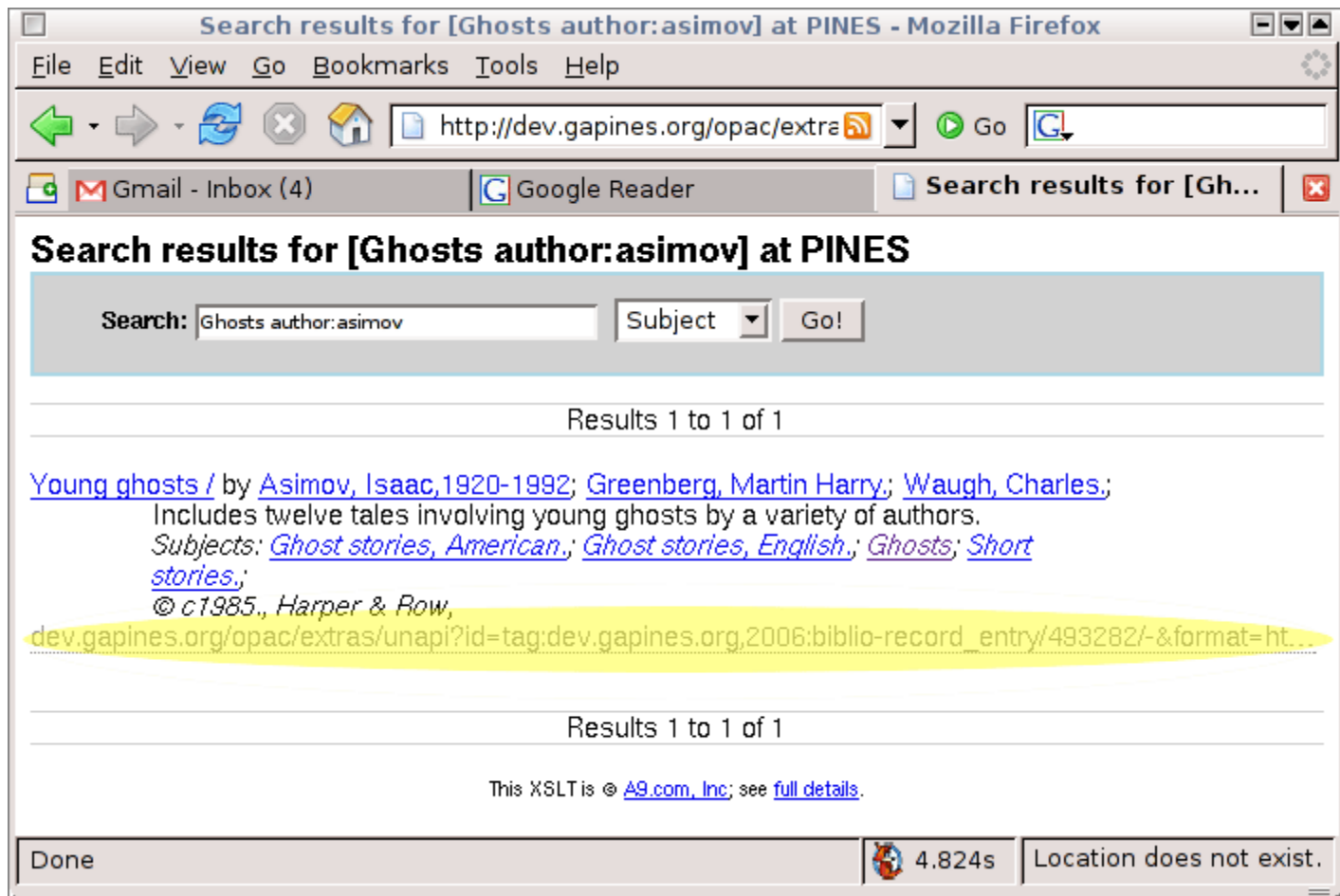


Figure 1: Screenshot of Evergreen Record

- University of Alberta: The University of Alberta Libraries are experimenting with a unAPI service that uses an SRU [24] proxy to retrieve records in various formats from the catalogue. This project is an example of the ease with which unAPI services can be layered on top of existing services. At the unAPI level, all that is required is an SRU server that can retrieve records using the "rec.id" field: a minimally-configured installation of IndexData's YAZ Proxy [25] (which provides an SRU interface in front of the catalogue's Z39.50 server) will do the trick. Our unAPI server is an Apache Cocoon installation with a set of custom pipelines that convert unAPI requests into SRU requests, retrieve the record, strip off the SRU wrapper, and return the record to the unAPI client. Another pipeline handles errors. The result is that with very little effort, and with no disruption of service to non-unAPI-aware clients, a unAPI service could be added to the catalogue. The biggest problem was determining how to embed the record id in a unAPI tag within our OPAC's templating system. The Cocoon code is freely available [26] and can be deployed in front of any SRU server; all it needs to know is the address of the SRU server and the record formats that can be requested.

Sample record view:

http://ualweb.library.ualberta.ca/uhtbin/cgiirsi/x/0/0/57/5?user_id=WUAARCHIVE&searchdata1=1565847547%7B020%7D

Sample unAPI formats list:

<http://chelsea.library.ualberta.ca/unapi/server?id=2623311>

Sample object via unAPI:

<http://chelsea.library.ualberta.ca/unapi/server?id=2623311&format=mods>

```
<map:match pattern="record">
  <map:generate src="
{global:sruServer}?version=1.1&amp;operation=searchRetrieve&amp;query=rec.id%3D{request-param:id}
&amp;maximumRecords=1&amp;recordSchema={request-param:format}" label="content"/>
  <map:transform src="xsl/unwrap.xsl" label="step1"/>
  <map:serialize type="xml"/>
</map:match>
<map:match pattern="recordformats" label="content">
  <map:generate src="xml/formats.xml"/>
  <map:transform src="xsl/formats2id.xsl" label="step1">
    <map:parameter name="id" value="{request-param:id}"/>
  </map:transform>
  <map:serialize type="xml" status-code="300" media-type="application/xml"/>
</map:match>
<map:match pattern="formats">
  <map:generate src="xml/formats.xml"/>
  <map:serialize type="xml"/>
</map:match>
```

Figure 2: Cocoon pipeline for SRU Proxy

- Canary Database: The Canary Database of animals as sentinels of human environmental health hazards collects article references from several abstracting and indexing services for additional indexing by researchers who classify studies of animal sentinels according to several criteria. Because the ability to save citations to a reference management tool is critical, the Canary Database provides easy links for users to download citations in a variety of useful formats. This feature, which supports export of a single record or sets of records in Pubmed, RIS, BibTeX, and MODS, is implemented via an unAPI interface. The links a user might click to export a record to a particular format are also unAPI links to those formats, so no distinction between the user and machine interfaces is necessary, simplifying code development and maintenance.

Sample record view:

<http://canarydatabase.org/record/488?view=export>

Sample unAPI formats list:

<http://canarydatabase.org/unapi?id=http://canarydatabase.org/record/488>

Sample object via unAPI:

<http://canarydatabase.org/unapi?id=http://canarydatabase.org/record/488&format=bibtex>

The screenshot shows a web browser window titled "Canary Database: Record 488". The address bar contains the URL <http://canarydatabase.org/record/488?view=e>. The page header features the Canary Database logo (a yellow bird inside a blue biohazard symbol) and the text "Canary Database Animals as Sentinels of Human Environmental Health Hazards". To the right, it says "Public Beta" and "Please contact us with your comments." Below the header is a search bar with a "go" button and a link to "Advanced Search".

The main content area is titled "Home" and displays the following information:

- Environ Sci Technol 2001 Apr 1;35(7):1339-42.
- Global distribution of perfluorooctane sulfonate in wildlife.**
- Giesy JP, Kannan K**
- Department of Zoology, National Food Safety and Toxicology Center, Institute for Environmental Toxicology, Michigan State University, East Lansing 48824, USA. jgiesy@aol.com
- [unapi:mods][unapi:ris][unapi:bibtex][unapi:endnote]
- Article type: Exposures only - Canary ID: 488

Below this information is a table with tabs for "Canary data", "Abstract", "Reference", "Related", and "Export". The "Export" tab is selected, showing the "Export this record" section. This section contains the text: "Export this record to your reference manager by clicking on the appropriate format below and using your web browser's 'file -> save as' menu function. Additional formats will be added as requested." Below this text is a list of export formats:

- BibTeX
- EndNote
- MODS
- RIS
- Tagged/Pubmed (NLM)

The footer of the page includes the text: "Yale University Occupational and Environmental Medicine Yale University School of Medicine Copyright 2004-2005, Yale University School of Medicine, New Haven, Connecticut, USA. All rights reserved. Please review our site disclaimer."

Figure 3: Canary Database record export links

unAPI with OpenURL Resolvers and Other Services

- Umlaut: The Umlaut OpenURL resolver at Georgia Tech provides unAPI services to OpenURL ContextObjects. Using a Key Encoded Value ContextObject as the identifier, the service analyses the incoming citation and aggregates appropriate metadata from other services such as link resolvers, catalogues, OAI-PMH repositories, Pubmed, Google, Yahoo and Connotea. For any given identifier it will return an XML-encoded

ContextObject and any of the above services that are applicable. It also will return an 'Umlaut document' which bundles all responding services in either XML or JSON (JavaScript Object Notation). The advantage here is that as the link resolver assumes an increasingly important role in bridging the library to the outside world, users will find they can enable unAPI services for any OpenURL-enabled resource. The fact that it uses COinS as its identifier lowers the barrier for resources that already support OpenURL to adopting unAPI.

Umlaut unAPI service:

<http://umlaut.library.gatech.edu/unapi?>

Sample unAPI formats list:

http://umlaut.library.gatech.edu/unapi?id=ctx_ver%3DZ39.88-2004%26ctx_enc%3Dinfo%253Aofi%252Fenc%253AUTF-8%26rft_id%3Dinfo%253Adoi%252F10.1038%252F438531a

Sample object via unAPI:

http://umlaut.library.gatech.edu/unapi?id=ctx_ver%3DZ39.88-2004%26ctx_enc%3Dinfo%253Aofi%252Fenc%253AUTF-8%26rft_id%3Dinfo%253Adoi%252F10.1038%252F438531a&format=umlaut-xml

- OPA: 'OPA Proxies APIs', or OPA for short, is a unAPI demonstration tool which proxies a variety of remote APIs through its own unAPI service. Its purpose is to show how some basic functions of otherwise incompatible third-party APIs may be quickly unified by unAPI. The latest revision of OPA proxies the Amazon, Flickr, and Pubmed APIs as well as arbitrary OAI-PMH services through a two-line-per-service configuration file. OPA provides several metadata and object format options for each of Amazon, Flickr, and Pubmed, and will proxy any available format from an OAI-PMH service. OPA also provides a simplistic object-wrapping format that uses JSON to package multiple object formats and metadata records into a single file, modelled after the MPEG21 DID object packaging strategy.

Sample Flickr.com record view:

<http://flickr.com/photos/dchud/31568800/>

Sample unAPI formats list:

<http://opa.onebiglibrary.net/?id=http://flickr.com/photos/dchud/31568800/>

Sample object via unAPI:

http://opa.onebiglibrary.net/?id=http://flickr.com/photos/dchud/31568800/&format=jpeg_Medium

unAPI in Weblogs

- WordPress plug-in: The unAPI plug-in for WordPress [27] provides records for each blog post and page in the following formats: OAI-Dublin Core , MODS, SRW-Dublin Core,

MARCXML, and RSS. WordPress, a PHP application, allows external applications and plug-ins to use its functions and gain access to information stored within a blog via standard PHP functions, therefore the plug-in was also coded in PHP.

- Technosophia: Technosophia, a blog about 'Libraries, Technology, and Infotainment,' provides an unAPI service via the WordPress plug-in. Posts are available in each format supported by the plug-in.

Sample blog view:

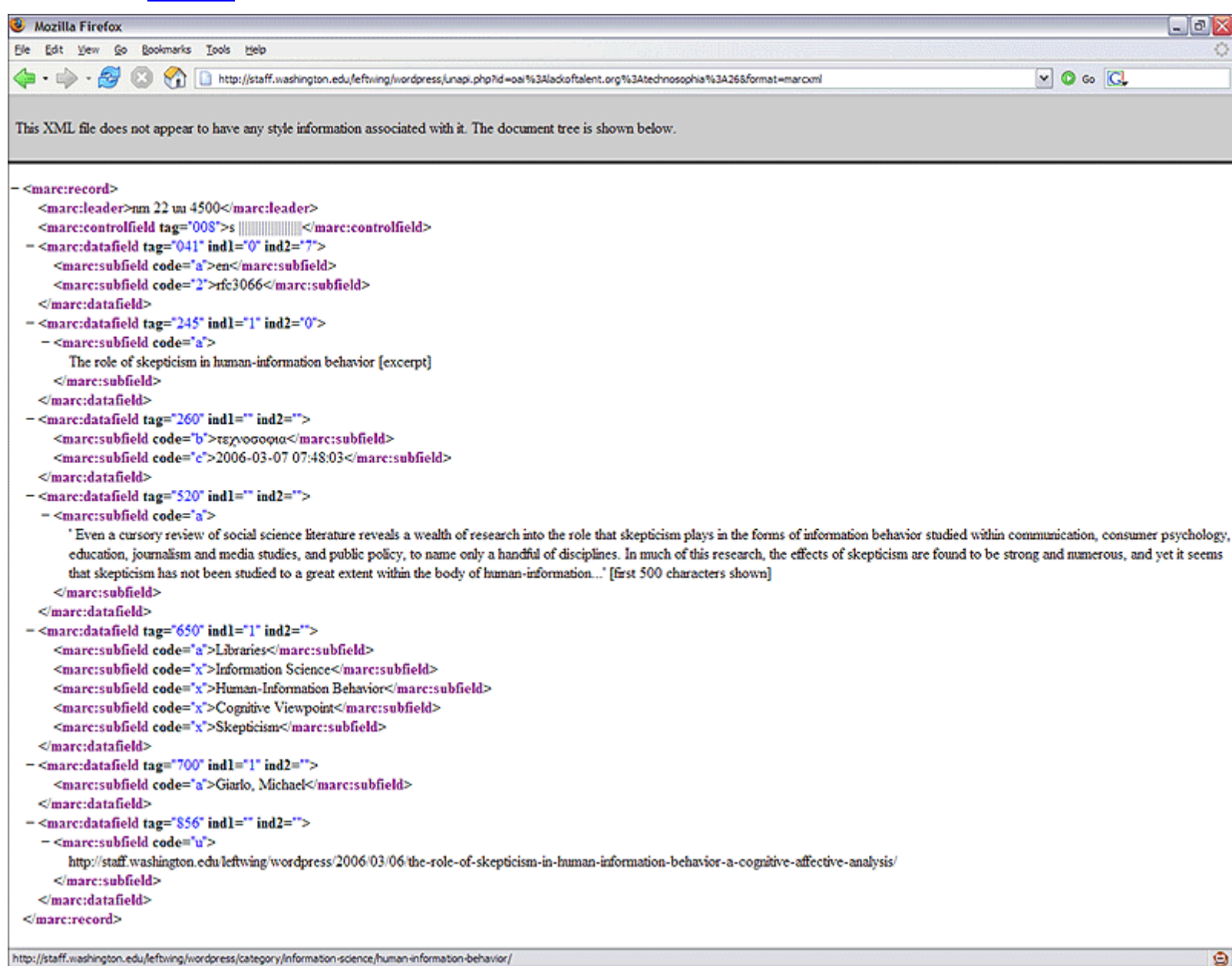
<http://lackoftalent.org/michael/blog/>

Sample unAPI formats list:

<http://lackoftalent.org/michael/blog/unapi.php?id=oai:lackoftalent.org:technosophia:45>

Sample object via unAPI:

<http://lackoftalent.org/michael/blog/unapi.php?id=oai:lackoftalent.org:technosophia:45&format=marcxml>



The screenshot shows a Mozilla Firefox browser window displaying an XML document tree for a MARCXML record. The address bar shows the URL: <http://staff.washington.edu/leftwing/wordpress/unapi.php?id=oai%3Alackoftalent.org%3Atechnosophia%3A26&format=marcxml>. The main content area displays the XML structure, including fields for leader, controlfields, datafields, and subfields. The datafields contain information such as the title 'The role of skepticism in human-information behavior [excerpt]', the author 'Giarto, Michael', and a URL to the full article.

```
- <marc:record>
  <marc:leader>nm 22 ua 4500</marc:leader>
  <marc:controlfield tag="008">s |||||</marc:controlfield>
  <marc:datafield tag="041" ind1="0" ind2="7">
    <marc:subfield code="a">en</marc:subfield>
    <marc:subfield code="2">rfc3066</marc:subfield>
  </marc:datafield>
  <marc:datafield tag="245" ind1="1" ind2="0">
    <marc:subfield code="a">
      The role of skepticism in human-information behavior [excerpt]
    </marc:subfield>
  </marc:datafield>
  <marc:datafield tag="260" ind1="" ind2="">
    <marc:subfield code="b">rs;voooot</marc:subfield>
    <marc:subfield code="c">2006-03-07 07:48:03</marc:subfield>
  </marc:datafield>
  <marc:datafield tag="520" ind1="" ind2="">
    <marc:subfield code="a">
      ' Even a cursory review of social science literature reveals a wealth of research into the role that skepticism plays in the forms of information behavior studied within communication, consumer psychology, education, journalism and media studies, and public policy, to name only a handful of disciplines. In much of this research, the effects of skepticism are found to be strong and numerous, and yet it seems that skepticism has not been studied to a great extent within the body of human-information...' [first 500 characters shown]
    </marc:subfield>
  </marc:datafield>
  <marc:datafield tag="650" ind1="1" ind2="">
    <marc:subfield code="a">Libraries</marc:subfield>
    <marc:subfield code="x">Information Science</marc:subfield>
    <marc:subfield code="x">Human-Information Behavior</marc:subfield>
    <marc:subfield code="x">Cognitive Viewpoint</marc:subfield>
    <marc:subfield code="x">Skepticism</marc:subfield>
  </marc:datafield>
  <marc:datafield tag="700" ind1="1" ind2="">
    <marc:subfield code="a">Giarto, Michael</marc:subfield>
  </marc:datafield>
  <marc:datafield tag="856" ind1="" ind2="">
    <marc:subfield code="u">
      http://staff.washington.edu/leftwing/wordpress/2006/03/06/the-role-of-skepticism-in-human-information-behavior-a-cognitive-affective-analysis/
    </marc:subfield>
  </marc:datafield>
</marc:record>
```

Figure 4: MARCXML from WordPress

- unapi.info: The news weblog for the unAPI specification is also a WordPress weblog with the unAPI plug-in installed.

Sample blog entry view:

<http://unapi.info/news/archives/16>

Sample unAPI formats list:

<http://unapi.info/news/unapi.php?id=http%3A//unapi.info/news/archives/16>

Sample object via unAPI:

http://unapi.info/news/unapi.php?id=http%3A//unapi.info/news/archives/16&format=oai_dc

Tools for Working with unAPI

Greasemonkey scripts: unAPI links are not necessarily visible to users by default. Client-side page rewriting scripts can provide a visual indicator of available unAPI links and formats where such indicators might not otherwise be present in page design. The following Greasemonkey scripts both provide visual indicators of available unAPI links using different styles; both are based on the same code.

- By Xiaoming Liu:
http://lxming.blogspot.com/2006_05_21_lxming_archive.html
- By Alf Eaton:
<http://cipolo.med.yale.edu/pipermail/gcs-pcs-list/2006-June/000951.html>

To use these scripts, first install Greasemonkey [16], restart your Firefox browser, and then install one or both of these scripts. Both will work on all of the sample record views listed above.

- The unAPI Ruby gem [30] provides an easy to install client library for interacting with unAPI services from Ruby. The client library also includes a command line version of the validation service.
- The unAPI Validator [31] supports implementers by providing instant feedback on how well a service implements the specification. It is a RubyOnRails application built on top of the unAPI Ruby gem, and it provides detailed warning recommendations and failure indicators in an easy-to-read summary table. A Validator bookmarklet is also available to make it even easier for implementers to test any page in their applications by sending any page open in a browser right into the validation suite.

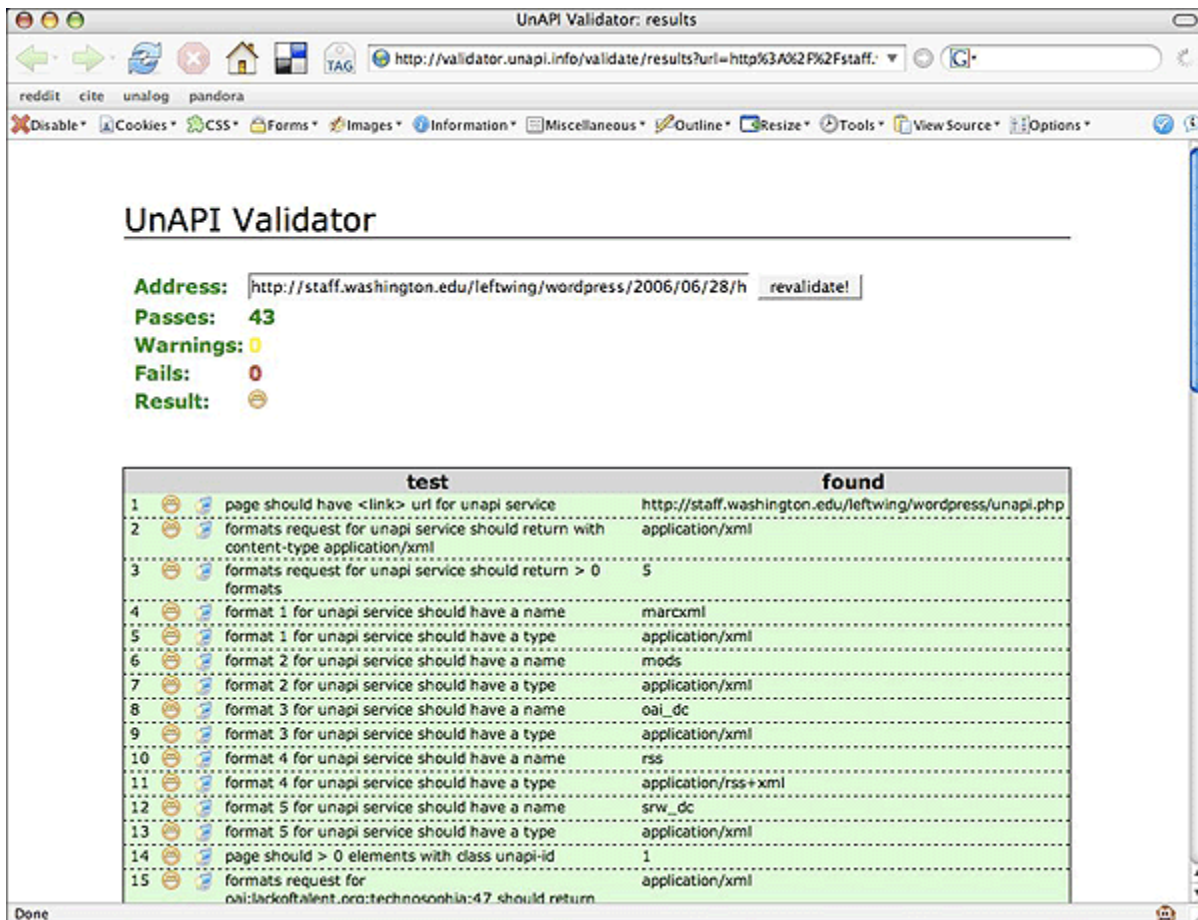


Figure 5: Web-based unAPI Validator

Highlights of Issues Considered

Like any other standards development process, the unAPI specification process had its share of difficult and contentious issues. Ultimately we chose the simplest solutions that would work in the widest possible set of applications. Some of these issues included:

- Use of identifiers: initially unAPI required URIs, but this was broadened to enable a wider range of resources to participate.
- Response format: the relative merits of plain text, JSON, and XML were debated, and the group chose a simplistic XML structure without namespaces or a schema.
- Microformats: participants in the unAPI process engaged the microformats.org community in discussion of a microformat for identifiers, but a consensus did not quickly emerge, so we followed the spirit and style of earlier microformats.
- Deferring to HTTP: we wavered on whether to specify HTTP status codes, finally agreeing to 'recommend' particular codes, again requiring less of publishers [28].
- Overlap with OpenURL / OAI-PMH: we felt that to meet the goal of broad adoption, imposing the complexity of OpenURL and the structure of OAI-PMH would be a losing proposition. unAPI could, however, serve as a 'gateway specification' to these protocols,

an easier bridge for general Web developers to cross when interacting with structured information objects.

Next Steps


With unAPI Version 1 complete, it is now available for general use. We have started to experiment by combining the unAPI copying functions with the Atom Publishing Protocol or with the Live Clipboard as a paste function. For example, we enhanced the unalog social bookmarking application to copy out objects found via unAPI in bookmarked pages, and to paste these objects in using Atom. Objects pasted into unalog are then available to other users in both the unalog user interface and through a new unAPI interface in unalog. Figure 6 demonstrates this; the images, from Flickr, were obtained through OPA's unAPI interface and its JSON object wrapping format.

unalog-SILDIN logged in as: [dchud](#) [[my](#)] [[entries](#)] [[tags](#)] [[groups](#)]

[Home](#) [People](#) [Tags](#) [Groups](#) [About](#) [Contact](#) [News](#) [Search](#)


Sunday, 2006-04-09

17:36 by [dchud](#) - [Gerduberg flora on Flickr - Photo Sharing!](#)




formats: [dc](#) [html](#) [jpeg_Medium](#) [jpeg_Original](#) [jpeg_Small](#) [jpeg_Square](#) [jpeg_Thumbnail](#) [mods](#)
[\[edit\]](#) [\[delete\]](#) with tag [flora](#)

17:31 by [dchud](#) - [Band Explosion on Flickr - Photo Sharing!](#)



formats: [dc](#) [html](#) [jpeg_Large](#) [jpeg_Medium](#) [jpeg_Original](#) [jpeg_Small](#) [jpeg_Square](#) [jpeg_Thumbnail](#) [mods](#)
[\[edit\]](#) [\[delete\]](#) with tag [aurora](#)

17:17 by [dchud](#) - [Legislature Aurora on Flickr - Photo Sharing!](#)



formats: [dc](#) [html](#) [jpeg_Large](#) [jpeg_Medium](#) [jpeg_Original](#) [jpeg_Small](#) [jpeg_Square](#) [jpeg_Thumbnail](#) [mods](#)
[\[edit\]](#) [\[delete\]](#) with tag [aurora](#)

Figure 6: unAPI Copy and Atom Paste in unalog.

As we move forward with implementing unAPI Version 1, we continue to watch related developing techniques such as microformats and HTTP header Link Templates [29]. If a microformat for identifying arbitrary identifiers in HTML or a similar technique within HTML itself emerged, unAPI would not need to specify use of the ABBR pattern. Similarly, if Link Templates or another technique made API patterns more easily discovered and specified, unAPI would not even need to define its own parameter names or LINK element semantics. If all of these missing pieces were to appear in widely accepted solutions, the unAPI object-copy paradigm could exist as a mere one-paragraph convention (a significant reduction from the present length of one and a half pages). In the meantime, we believe that unAPI Version 1 can

help to get more out of library - or any other - Web applications. It follows the Unix traditions of doing one thing well and being easily connected and combined to form more complex functionality. We hope that it proves useful and helps to bring the library community closer to the level of simplified integration demanded by users today.

References

1. van Veen, T., "Serving Services in Web 2.0", *Ariadne* 47, April 2006
<http://www.ariadne.ac.uk/issue47/vanveen/>
2. Chudnov, D., Frumkin, J., Weintraub, J., Wilcox, M., Yee, R., "Towards Library Groupware with Personalised Link Routing", *Ariadne* 40, July 2004
<http://www.ariadne.ac.uk/issue40/chudnov/>
3. Rhyno, A., "WebDAV and Lessons from Blackfoot Physics", LibraryCog weblog, February 19, 2004
<http://librarycog.uwindsor.ca:8087/artblog/librarycog/1077044415/1077205713/1077219993>
4. Burcham, B., "Baby Steps to Synergistic Web Apps", lesscode weblog, October 21, 2005
<http://lesscode.org/2005/10/21/baby-steps-to-synergistic-web-apps/>
5. Ozzie, R., "Wiring the Web", Ray Ozzie's weblog, March 7, 2006
<http://rayozzie.spaces.msn.com/blog/cns!FB3017FBB9B2E142!285.entry>
6. Ozzie, R. et al., "Live Clipboard Technical Introduction", MSN Spaces, March 2006
<http://spaces.msn.com/editorial/rayozzie/demo/liveclip/liveclipsample/techPreview.html>
7. Microformats project home, <http://microformats.org/>
8. Van de Sompel, H., et al., "Augmenting Interoperability: Presentations", Andrew W. Mellon Foundation wiki, April 25, 2006,
<http://msc.mellong.org/Meetings/Interop/presentations>
9. Bekaert, J., Hochstenbach, P., Van de Sompel, H., "Using MPEG-21 DIDL to Represent Complex Digital Objects in the Los Alamos National Laboratory Digital Library", *D-Lib Magazine* 9(11), November 2003,
<http://www.dlib.org/dlib/november03/bekaert/11bekaert.html>
10. Fedora Project, "Fedora Digital Objects: Fedora Release 2.1.1", April 4, 2004,
<http://www.fedora.info/download/2.1.1/userdocs/digitalobjects/>
11. Lagoze, C., Van de Sompel, H., Nelson, M., Warner, S., "The Open Archives Initiative Protocol for Metadata Harvesting: Protocol Version 2.0", June 14, 2002,
<http://www.openarchives.org/OAI/openarchivesprotocol.html>
12. Van de Velde, E.F. (chair), et al., "ANSI/NISO Z39.88-2004: The OpenURL Framework for Context-Sensitive Services", NISO Press, April 15, 2005
http://www.niso.org/standards/standard_detail.cfm?std_id=783
13. Hellman, E., editor, "OpenURL ContextObjects in SPANs (COinS): stable version 1.0", August 8, 2005, <http://ocoins.info/>
14. gcs-pcs-list, Internet Email List, <http://cipolo.med.yale.edu/mailman/listinfo/gcs-pcs-list>
15. "Icons: It's still orange", Microsoft Team RSS Blog, December 14, 2005,
<http://blogs.msdn.com/rssteam/archive/2005/12/14/503778.aspx>
16. Boodman, A., Dunck, J., "Greasemonkey", Firefox browser extension,
<http://greasemonkey.mozdev.org/>

17. Chudnov, D., "ROGUE #1 - COinS-PMH", dchud's work log, September 27, 2005, <http://curtis.med.yale.edu/dchud/log/project/rogue/rogue-no.1-coins-pmh>
18. Chudnov, D., "Photos tagged with 'coinspmh'", Flickr.com, <http://flickr.com/photos/dchud/tags/coinspmh>
19. Van de Sompel, H., Nelson, M.L., Lagoze, C., Warner, S., "Resource Harvesting within the OAI-PMH Framework", *D-Lib Magazine*, December 2004, <http://www.dlib.org/dlib/december04/vandesompel/12vandesompel.html>
20. Chudnov, D., et al. "unAPI: An un-API for Webapps", June 23, 2006, <http://unapi.info/>
21. Chudnov, D., et al. "unAPI Version 1", June 23, 2006, <http://unapi.info/specs/unapi-version-1.html>
22. A9.com, "OpenSearch", <http://opensearch.a9.com/>
23. Nottingham, M., Sayre, R., editors, "RFC 4287: The Atom Syndication Format", December 2005, <http://rfc.net/rfc4287.html>
24. The Library of Congress, "SRU: Search and Retrieve via URL", May 30, 2006, <http://www.loc.gov/standards/sru/>
25. IndexData ApS, "YAZ Proxy", 2006, <http://www.indexdata.dk/yazproxy/>
26. Binkley, P., "unAPI over SRU with Cocoon", Quaedam cuiusdam weblog, July 2006, <http://www.wallandbinkley.com/quaedam/?p=69>
27. Giarlo, M., Binkley, P. "unAPI WordPress Plug-in", Technosophia weblog, June 2006, <http://lackoftalent.org/michael/blog/unapi-wordpress-plug-in/>
28. Dodds, L., "Connecting Social Content Services using FOAF, RDF, and REST", XTech 2005 Conference, <http://www.idealliance.org/proceedings/xtech05/papers/02-07-04/>
29. Nottingham, M., "HTTP Header Linking Internet Draft", June 16, 2006, <http://www.ietf.org/internet-drafts/draft-nottingham-http-link-header-00.txt>
30. Summers, E., "unapi", textualize Web site, July 2006, <http://www.textualize.com/unapi>
31. Summers, E., "unAPI Validator", July 2006, <http://validator.unapi.info/>

Author Details

Daniel Chudnov

Librarian/Programmer
Yale Center for Medical Informatics

Email: daniel.chudnov@yale.edu

Web site: <http://onebiglibrary.net/>

Peter Binkley

Digital Initiatives Technology Librarian
University of Alberta Libraries

Email: peter.binkley@ualberta.ca

Web site: <http://www.wallandbinkley.com/quaedam/>

Jeremy Frumkin

The Gray Family Chair for Innovative Library Services
Oregon State University Libraries

Email: jeremy.frumkin@oregonstate.edu
Web site: <http://digitallibrarian.org>

Michael J. Giarlo
Senior Computer Specialist
University of Washington Libraries

Email: leftwing@alumni.rutgers.edu
Web site: <http://purl.org/net/leftwing/blog>

Mike Rylander
Database Developer
Georgia Public Library Service - PINES Development

Email: mrylander@gmail.com
Web site: <http://open-ils.org/>

Ross Singer
Application Developer
Georgia Tech Libraries

Email: ross.singer@library.gatech.edu
Web site: <http://rsinger.library.gatech.edu/bio/>

Ed Summers
Software Developer
CIAC

Email: ehs@pobox.com
Web site: <http://textualize.com/>

[Return to top](#)

Article Title: "Introducing unAPI"
Author: Daniel Chudnov, Peter Binkley, Jeremy Frumkin, Michael J. Giarlo, Mike Rylander, Ross Singer and Ed Summers
Publication Date: 30-July-2006 Publication: Ariadne Issue 48
Originating URL: <http://www.ariadne.ac.uk/issue48/chudnov-et-al/>

This article has been published under copyright; please see our [access terms and copyright](#) guidance regarding use of content from this article. See also our explanations of [how to cite Ariadne articles](#) for examples of bibliographic format.

Ariadne is published by [UKOLN](#). UKOLN receives support from [JISC](#) and the [University of Bath](#) where it is based.

© Ariadne ISSN: 1361-3200. See our explanations of [Access Terms and Copyright](#) and [Privacy Statement](#).

Source URL (retrieved on *17 Jul 2013 - 22:15*): <http://www.ariadne.ac.uk/issue48/chudnov-et-al>