Kepler – A Communal Digital Library

K. Maly[1], M. Nelson[1], M. Zubair[1], A. Amrou[1], S. Kothamasa[1], L. Wang[1], R. Luce[2]

[1]Old Dominion University Computer Science Department, Norfolk VA 23529 USA
{maly,mln,zubair,aamr,kumar_s,wang_l}@cs.odu.edu
[2]Los Alamos National Laboratory, Research Library, Los Alamos NM 87544 USA
rick.luce@lanl.gov

Abstract. Kepler is an attempt to bridge the gap between established, organization-backed digital libraries and groups of researchers that wish to publish their findings under their control, anytime, anywhere yet have the advantages of an OAI-compliant digital library. We describe an architecture and implementation of the Kepler system that allows an archivelet to be installed in the order of minutes by an author on a personal machine and a group server in less than an hour. The group server will harvest from all archivelets and make the union of all published papers available for search to a community. We describe how a group administrator can provide an XML schema for the metadata and how the Kepler engine will validate against them when an author publishes a paper and completes the metadata. We have demonstrated that we can surmount the technical difficulties for authors to publish as easy as to a website yet produce OAI-compliant digital libraries.

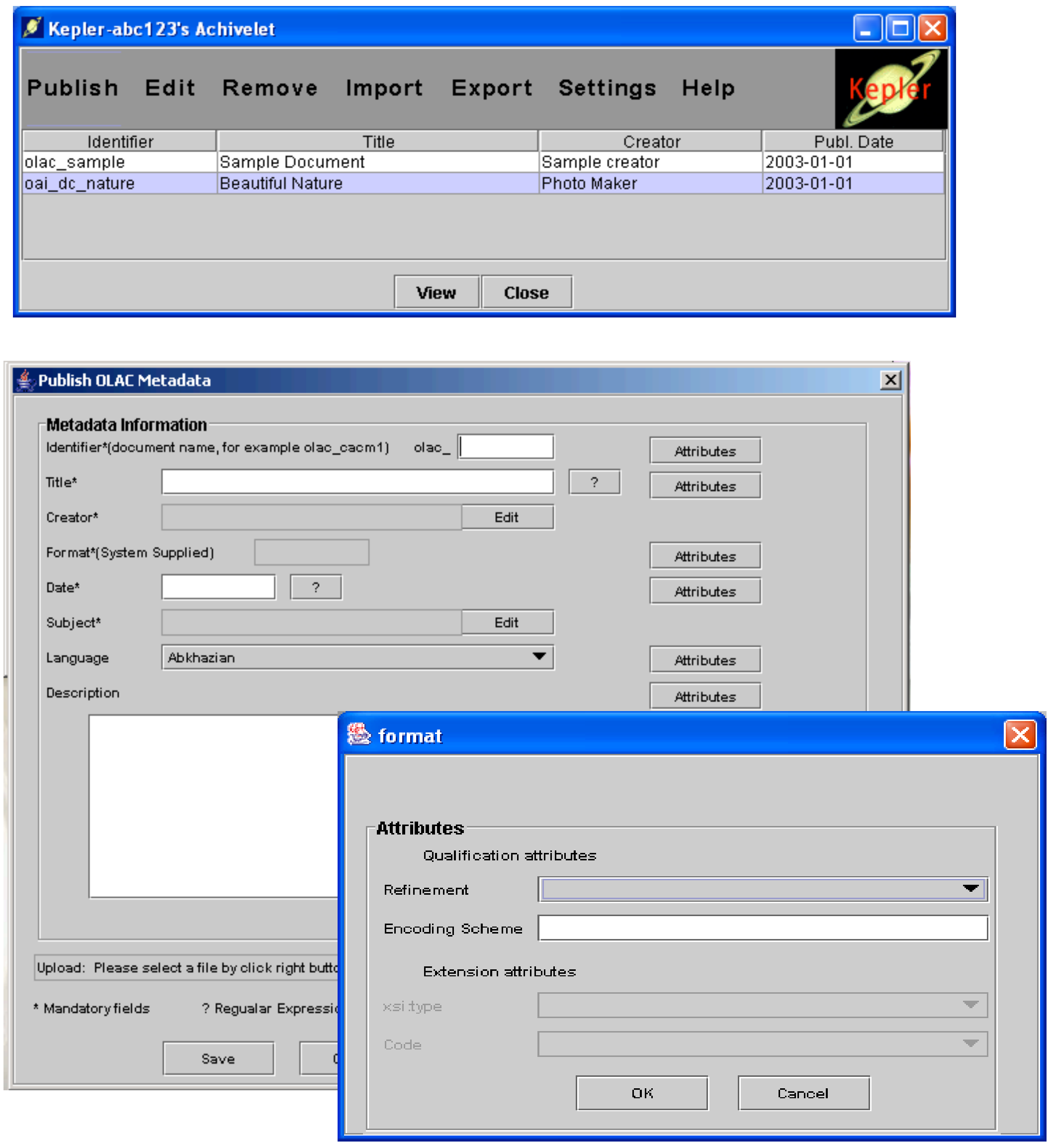
1 Introduction

One of the largest obstacles for information dissemination to a user community is that many digital libraries use different, proprietary technologies that inhibit interoperability. The Open Archives Initiative (OAI) addresses interoperability by using a framework to facilitate the discovery of content stored in distributed archives through the use of the Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH) [1]. Realizing the benefits of OAI, a number of communities are interested in an out-of-the-box solution that will help them deploy OAI-based digital libraries. However, building a communal digital library is currently severely hampered by the lack of easy to use tools that address the diverse requirements of different communities. In particular, metadata, as the codification of the worldviews that define a community, needs to accommodate varying formats, uses, encodings and pedigrees. Creating a system for communal digital libraries poses a number of challenging research questions:

One of our initial efforts in this direction is Kepler [2], which gives publication control to individual publishers, supports rapid dissemination, and addresses interoperability. In Kepler, OAI-PMH is used to support "personal data providers" or "archivelets". Archivelets are meant to be “personal pocket libraries” that are on the one hand OAI-PMH compliant data providers, and on the other hand overcome the reluctance of authors to publish into a digital library (instead of putting their work on their website) through user-friendly publishing tools and total control through having all files and metadata reside on the author's personal machine. This latter characteristic has some serious implication on the reliability issue since these machines are not necessarily always up or connected to the Internet (e.g. the author might keep the archivelet on a laptop and be on travel or on a field trip). As a demonstration, we provided an initial registration service and a service provider at Old Dominion University. Once an archivelet registered with our registration service, the service provider could harvest metadata from it. We faced a number of issues during the initial deployment of Kepler: the software did not have the flexibility of customizing and deploying it for a community, and archivelets were often installed behind NATs making it difficult for the service provider to harvest them. In this paper, we build upon our experiences with the initial Kepler distribution and describe tools and software for groups within communities to deploy digital libraries that are customized for their needs, easily populated, managed, and "open" for development of future services. The main contributions of this paper are: (a) a framework for defining, describing and supporting the publication/dissemination requirements of a community group; (b) packaged tools and software to create an out-of-the-box solution for deploying communal digital libraries for diverse groups; (c) for use behind firewalls, we have developed a server-based publication tool that mirrors the archivelets in terms of functionality. A short summary of part of this work has been accepted in [3]. For the test deployment, we are working with the US Geological Survey (USGS), Los Alamos National Laboratory (LANL), and the Open Language Archives Community (OLAC).

2 Modular Architecture

The original archivelet implementation [2] only supported Dublin Core (DC) format. We remedied this problem by allowing an arbitrary number of formats to be realized by one or more communities. We have re-engineered the architecture to be extensible with regard to formats and functions (See Figure 2). The new design has a well-defined API specification defining the various functions that are implemented by every module that in turn are available for other modules. Support for new metadata formats requires just the implementation of a metadata driver module. In the Kepler software documentation, we provide developer guidelines for fast and easy implementation of a metadata driver. The metadata manager module is responsible for instantiating the various metadata drivers the system is configured to use. It also implements the OAI-PMH API that provides a method for each of the six OAI-PMH requests. OAI-PMH requests received by the Webserver module are forwarded to the Driver Manager that decides what metadata drivers are involved and invokes these drivers to get partial responses from each. The Driver Manager then constructs the whole response from these partial responses. The Driver Manager also implements a User Interface API. This API contains methods that are invoked in response to user interactions with the main interface. For example, when the user clicks "publish", the Driver Manager brings a simple GUI that allows the user to select which metadata format she wants to use and then the Driver Manager invokes the appropriate Driver to display the appropriate Publishing tool (see Figure 1).


Kepler-abc123's Achivelet
Publish Edit Remove Import Export Settings Help
Kepler
View Close
Publish OLAC Metadata
Edit
?
Attributes
Save
format
OK Cancel

Identifier	Title	Creator	Publ. Date
olac_sample	Sample Document	Sample creator	2003-01-01
oai_dc_nature	Beautiful Nature	Photo Maker	2003-01-01

Metadata Information
Identifier*(document name, for example olac_cacm1) olac_
Title*
Creator*
Format*(System Supplied)
Date*
Subject*
Language: Abkhazian
Description
Upload: Please select a file by click right butto
* Mandatory fields
? Regualar Expressio

Attributes
Qualification attributes
Refinement
Encoding Scheme
Extension attributes
xsi:type
Code

Fig. 1. Main Interface and OLAC Publishing Tool for Archivelets

The metadata driver module implements the OAI-PMH processing and the user interface functions such as publishing tools for the specific metadata format that the Driver handles. The publishing interface (see bottom left of Figure 1) has dynamic field types (mandatory or optional), which are determined by a configuration file, based on XML schema, managed by the group server administrator. The Driver invokes the Validation module whenever new metadata is published to validate the metadata against the constraints specified in the configuration file and uses the repository API to store metadata and files.

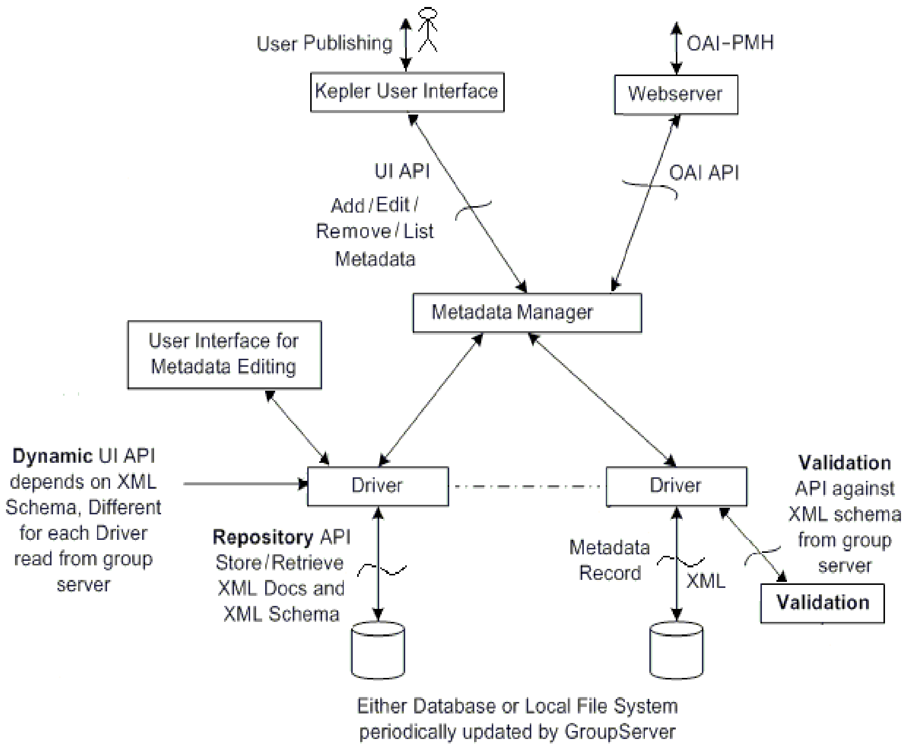
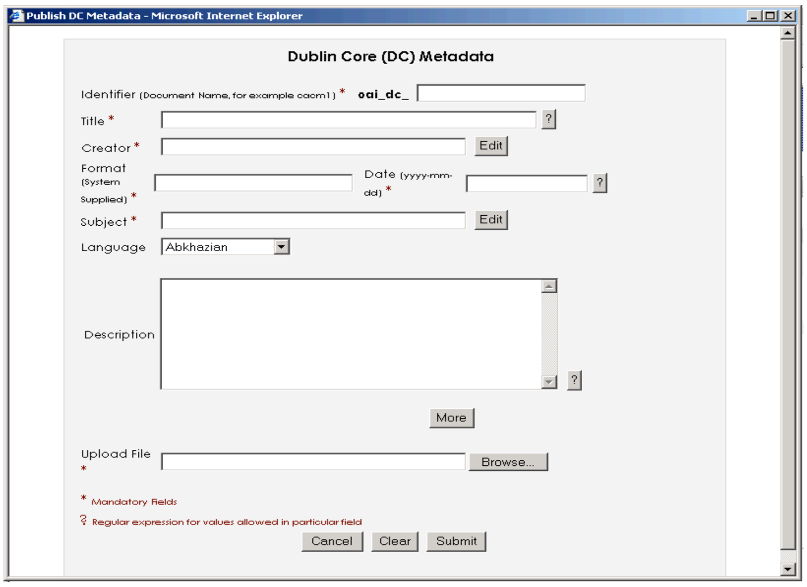

User Publishing
OAI-PMH
Kepler User Interface
Webserver
UI API
Add / Edit / Remove / List Metadata
OAI API
Metadata Manager
User Interface for Metadata Editing
Dynamic UI API depends on XML Schema, Different for each Driver read from group server
Driver
Driver
Validation API against XML schema from group server
Repository API Store / Retrieve XML Docs and XML Schema
Metadata Record
XML
Validation
Either Database or Local File System periodically updated by GroupServer

Fig. 2. Archivelet Architecture

Server-side Architecture: We discovered many situations that required a server-side solution. Such examples include when the firewall is not controllable by the author, or when the author is on travel to a different organization, or when the author's organization has a strict security policy, or when only Internet access is available but not the personal laptop with the archivelet. We sometime refer to it as Internet café publishing. The main advantage of the server-side archivelet is that it can be accessed from anywhere by either humans or harvesters (OAI-PMH).

The Kepler Server-side archivelet can be accessed from any Kepler GDL and is governed by the validation rules specified by that GDL. The functionality as the user sees it is identical, thus it supports publishing metadata in DC and OLAC formats, creates persistent URLs for each individual archivelet, and allows any service provider (e.g. GDLs) to harvest metadata and allows users to import/export metadata.


Publish DC Metadata - Microsoft Internet Explorer

Dublin Core (DC) Metadata

Identifier (Document Name, for example cacm1) * oai_dc_

Title *

Creator *

Format (System Supplied) *

Date (yyyy-mm-dd) *

Subject *

Language: Abkhazian

Description

Upload File *

* Mandatory Fields

? Regular expression for values allowed in particular field

Fig. 3. DC Publishing Tool for Server-side Archivelet

The server-side archivelet system can be logically divided into 3 layers as shown in Figure 4: View (User Interface), Controller, and Model. Comparing to Figure 2, it can be seen that the basic, extensible driver architecture is the same, though not the implementation; the difference lies in the user interaction modules and the interaction with the GDL. In the server-side architecture, the archivelet resides on the GDL. The user interface layer provides various user interfaces for authors to create an archivelet, publish metadata and perform the other activities defined in the main archivelet interface (see Figure 3). The controller layer has several components (servlets and metadata drivers) that control the sequence of transitions between user interfaces, flow of data between user interfaces and the database, as well as handle OAI-PMH requests. The model layer performs the actual database operations. The key component of this layer is the DB Repository. It has the capability to handle metadata regardless of its native format. It provides database operations that are required for archivelet registration, configuration, saving and extracting metadata.

3 New Features

Persistent URLs: Archivelets are roamers; they can be at different locations with different IP addresses at different times. One of the metadata items in each published record in the archivelet is the URL of the document associated with the metadata. The document is stored on the machine the archivelet resides. Clearly we have a problem with using an archivelet's current IP addresses as part of the URL. We use instead persistent URLs for archivelets and their metadata records and their documents. In prior implementations every archivelet had a base URL of the format:

http://machinename.or.ip[:port]/OAIRequestHandlerScript

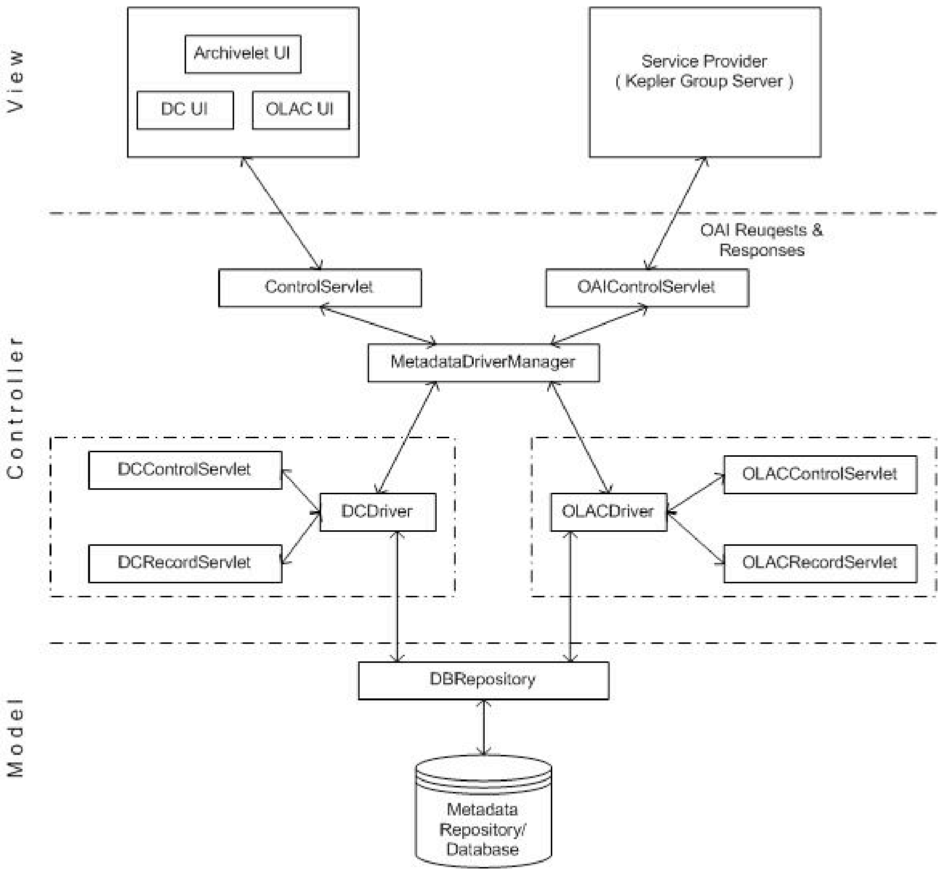


Fig. 4. Server-side Archivelet Architecture

This base URL was distributed to service providers (e.g. Kepler group servers), so they could issue OAI-PMH requests to harvest metadata from the Archivelet. It is possible that an archivelet can be installed on a machine whose IP address changes every time it connects to a network. In this scenario, if the change in an archivelet's base URL is not updated at the service providers, they cannot contact the archivelet to harvest metadata. A persistent URL for an archivelet is a base URL that is independent of archivelet's IP address. . The archivelet distributes this persistent URL to service providers instead of its actual base URL.

To use an archivelet for publishing metadata, it has to be registered with a Kepler GDL. While registering, the archivelet sends its actual base URL, along with other information, to the group server. On successful registration, the group server creates a persistent URL of the format

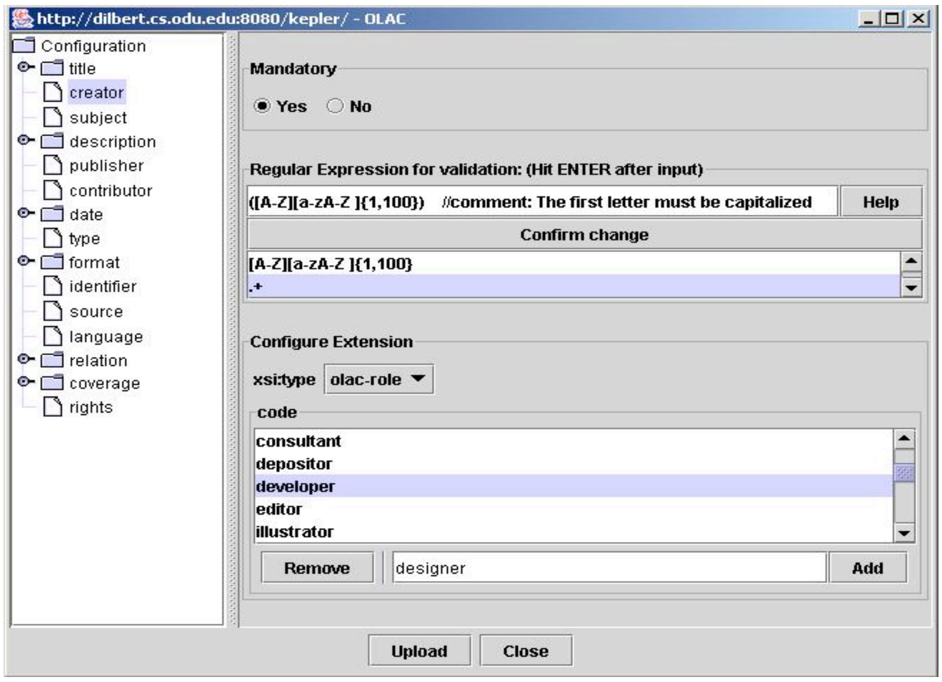
```
http://group.server.url[:port]/path-to-
OAIControlServlet/tr/archivelet-name

http://kepler.cs.odu.edu/testgroup/servlet/OAIControlSe
rvlet/tr/maly
```

The group server stores a mapping between an archivelet's actual base URL and its persistent URL. Also, it sends this persistent URL to the archivelet as a response to the registration request. To ensure that the persistent URL always maps to the most current base URL, the archivelet sends its current base URL to the group server at a predefined interval (e.g. on every startup) and the group server updates the persistent URL mapping for that archivelet.

When an OAI-PMH request is issued to this persistent URL, the OAIControlServlet on the appropriate Kepler GDL gets the request. The OAIControlServlet parses this request into two parts: the persistent URL and the OAI-PMH request (verb and parameters). Using the persistent URL it looks up its mapping table to resolve the actual base URL of the archivelet. Once it obtains the actual base URL, it appends the OAI-PMH request and directs the request to the archivelet. The archivelet responds to the OAI-PMH request and the OAIControlServlet redirects it to the actual requestor. Thus the persistent URL masks the dynamic nature of an archivelet's base URL.

Validation Tool: One defining feature of a community is their publication process and the level of associated control. The minimal requirement that Kepler institutes is the use of DC metadata. However, DC in itself has no mandatory fields and it is up to a community to define the semantics of a field and how it is to be used. This motivates us to adopt a process that can be described, and more importantly, enforced. The Validation module serves two purposes: (1) it provides declarations and implementations of a set of Validation APIs, see figures 2 and 4, that can be called by drivers (or any other Kepler modules) for metadata validation; (2) it provides a set of tools to facilitate the driver developer to generate an administration GUI for a new metadata set. The motivation of such an administration GUI is to give the group administrator some flexibility to tailor a general metadata set to an enforceable set of guidelines. For example, some groups may require that the creator field of an OLAC metadata record published in this group is mandatory and must begin with a capitalized letter. Such constraints are not documented in the published OLAC XML schemas. By using the administration GUI, a group administrator can impose these constraints conveniently in her group without directly editing the OLAC schema files. Figure 5 shows the administration GUI while tailoring an OLAC metadata set.

The current implementation of the Validation module allows a group administrator to: (1) specify whether a field is mandatory or optional; (2) specify regular expressions for a field (for example, the date field must be in the format yyyy-mm-dd); (3) add/delete items in the option list for some specific fields (for example, the value of language field can only be chosen from "dz", "el", "en", "eo", "es"). The validation module is designed to work from an XML schema describing a metadata set. In this implementation we assume that the XML Schemas for any new metadata set always will follow the authoring style of DC and OLAC schemas in defining and qualifying elements by means of Element Refinement and Encoding Schemes. The Validation module depends on these authoring styles as useful heuristics when extracting initial configuration from the original metadata XML Schemas. The initial configuration is extracted, then stored in an intermediate configuration file, which maps an XML namespace and element/type/attribute pair to its mandatory/option setting, regular expression and option list documented in the original schemas. The subsequent changes on the configuration that a group administrator makes through the administration GUI are written into this intermediate configuration file, without polluting the original metadata schemas.


http://dilbert.cs.odu.edu:8080/kepler/ - OLAC

Configuration: title, creator, subject, description, publisher, contributor, date, type, format, identifier, source, language, relation, coverage, rights

Mandatory: [✓ Yes] No

Regular Expression for validation: (Hit ENTER after input)

([A-Z][a-zA-Z]{1,100}) //comment: The first letter must be capitalized

[A-Z][a-zA-Z]{1,100}
.+

Configure Extension

xsi:type: olac-role

code: consultant, depositor, developer, editor, illustrator

designer

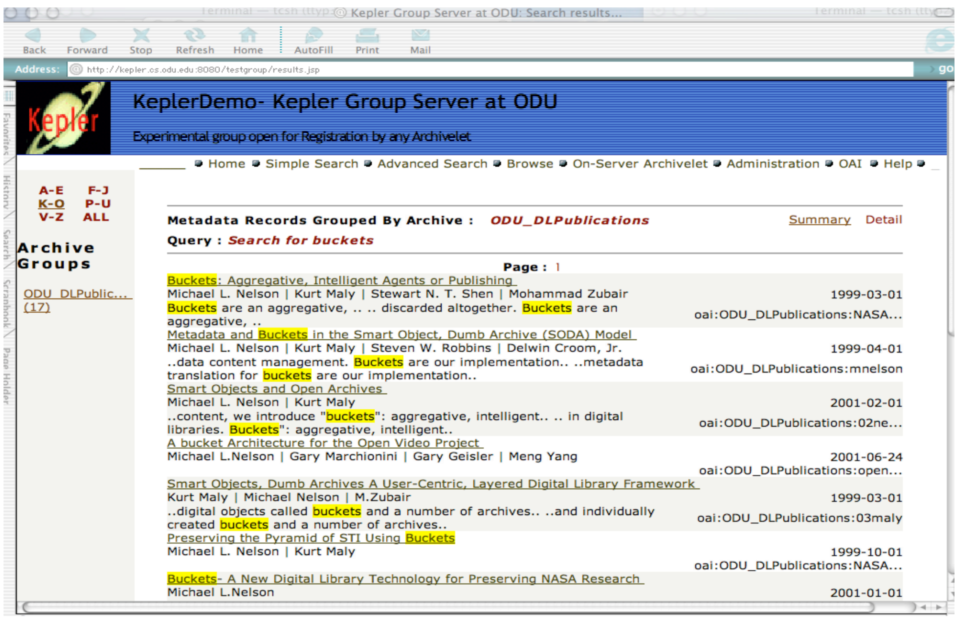
Fig. 5. An example administration GUI for OLAC

As part of a new metadata driver, the administration GUI needs to be implemented by the driver developer. To facilitate this process, the Validation module introduces an additional layer of indirection – GUI configuration file. A GUI configuration file is actually an XML instance file controlling the tree structure displayed on the left panel of the administration GUI. Specifically, each element in this XML file serves to map between a tree node in the GUI and an element/type/attribute defined in the metadata schemas. By editing this file to specify fields that are eligible for tailoring by group administrator, a driver developer can make use of the tools provided by the Validation module to generate the administration GUI automatically.

NAT/Proxies: Network Address Translator (NAT) Proxies are issues specific to the archivelet as it runs a server that listens for OAI-PMH and file download requests. The existence of a NAT might cause the communications between the harvester and the OAI-PMH server in the archivelet to fail. For handling NAT/Proxy issues, we use port mapping (also called port forwarding). It requires that the user configure the

NAT/Proxy device or software to forward incoming communications on some port to the archivelet software.

Import/Export: From our own internal use of archivelets, it became clear that there is a need for archivelet information exchange. Although archivelets are harvestable from the outside, they did not originally have the capability to receive information (OAI-PMH does not support two-way traffic). We decided to adopt OAI–PMH Static Repository [4]. The static repository specifies an XML schema that contains all the necessary information for OAI-PMH "frozen" in a single file. The use of a Static Repository Gateway allows a harvester to harvest a static repository as if it were a regular OAI-PMH data provider. Since in Kepler the archivelet typically contains less than 100 records, it is easy to transmit all the information in the OAI-PMH Static Repository format. This is done with the export/import feature in Kepler. It enables the user to export the metadata in her collection to a file, which can then be shared with others directly to avoid re-entry of metadata again and again. A typical scenario is when an author has several co-authors and can share the metadata with them, having them avoid entering the information into their archivelets. This does raise though the issue of duplication at the group server. In this implementation we have not addressed the duplication issue; duplicate records will simply be listed multiple times as coming from different archivelets.



http://kepler.cs.odu.edu:8080/testgroup/results.jsp

KeplerDemo- Kepler Group Server at ODU

Experimental group open for Registration by any Archivelet

Metadata Records Grouped By Archive : ***ODU_DLPublications***

Query : ***Search for buckets***

Page : 1

Buckets: Aggregative, Intelligent Agents or Publishing
Michael L. Nelson | Kurt Maly | Stewart N. T. Shen | Mohammad Zubair — 1999-03-01
Buckets are an aggregative, ... discarded altogether. Buckets are an aggregative, .. — oai:ODU_DLPublications:NASA...

Metadata and Buckets in the Smart Object, Dumb Archive (SODA) Model
Michael L. Nelson | Kurt Maly | Steven W. Robbins | Delwin Croom, Jr. — 1999-04-01
..data content management. Buckets are our implementation.. ..metadata translation for buckets are our implementation.. — oai:ODU_DLPublications:mnelson

Smart Objects and Open Archives
Michael L. Nelson | Kurt Maly — 2001-02-01
..content, we introduce "buckets": aggregative, intelligent.. .. in digital libraries. Buckets": aggregative, intelligent.. — oai:ODU_DLPublications:02ne...

A bucket Architecture for the Open Video Project
Michael L.Nelson | Gary Marchionini | Gary Geisler | Meng Yang — 2001-06-24
oai:ODU_DLPublications:open...

Smart Objects, Dumb Archives A User-Centric, Layered Digital Library Framework
Kurt Maly | Michael Nelson | M.Zubair — 1999-03-01
..digital objects called buckets and a number of archives.. ..and individually created buckets and a number of archives.. — oai:ODU_DLPublications:03maly

Preserving the Pyramid of STI Using Buckets
Michael L. Nelson | Kurt Maly — 1999-10-01
oai:ODU_DLPublications:NASA...

Buckets- A New Digital Library Technology for Preserving NASA Research
Michael L.Nelson — 2001-01-01

Fig. 6. Results from the Simple Search

Search Service and Caching: The group server offers a search service on the metadata harvested from the Kepler archivelets (and other OAI-PMH data providers). The search service is based on the Arc search engine [5, 6]. The group server offers a simple search interface that searches all metadata fields, an advanced search interface that allows fielded searching, and a browsing interface for extemporaneous resource discovery. Using the demonstrator group server in use at the Old Dominion University Computer Science Department, Figure 6 shows the metadata records resulting from a query. The group server performs caching by default. Clicking on the title in a record (Figure 6) displays all the metadata fields of that record in a separate window. The original resource is available in the DC.Identifier field, for example: http://128.82.7.77:2048/EDICTfinal0913.doc

However, the URL of the original resource may not always be available. Although the original URL is always presented, the GDL will also preemptively cache the resource at the GDL. For example, the URL of the above source is also available at:

http://kepler.cs.odu.edu:8080/testgroup/cache/oai.ODU_DLPublications.EDICTfinal0913.doc

Preemptively caching the resources increases the availability of the resources and insulates the GDL from the varying accessibility of the archivelets.

5 Related Work

The version of Kepler described in this paper draws from a significant base of existing OAI-PMH projects, developed both at Old Dominion and throughout the community. In addition to the original Kepler project [2], some of the features draw from the Arc [5, 6] and Archon [7, 8] projects. As mentioned above, the OAI-PMH Static Repository format [4] was adopted when the need for archivelet importing and exporting was addressed.

In the emerging field of "institutional repository software", many open source entries have emerged. The eprints.org software [9] from the University of Southampton has been widely adopted. CDSWare [10] was created at CERN and has been adopted in other locations as well. DSpace [11] is created by Hewlett Packard and the MIT Libraries, but has been widely adopted outside of MIT. All of these systems feature significant features and capability for building large-scale digital libraries and institutional repositories. All also use the OAI-PMH as a core technology. However, they are in contrast to Kepler in that they require significant resources to establish and maintain; the installation of these systems falls significantly beyond the 10-minute target of a Kepler GDL. The choice of a community of institutional digital library is an extremely important one, and we recommend surveys such as [12], [13] or [14] as guides in determining which DL suits your needs.

The persistent URL work is similar to the Extensible Repository Resource Locators (ERRoLs) project currently underway at OCLC [15], and which was first described as "Partial PURL Redirects" in [16]. ERRoLs, and its predecessor, describe a way to attach persistent, "human-friendly" URLs to OAI-PMH repositories and metadata objects inside those repositories. Our approach differs in that we focus only on repository naming, and is CDL/GDL-centric – as opposed to centric to a registry at OCLC or UIUC.

6 Conclusions and Future Work

We have done extensive use testing of both the installation process and the publishing tasks. The testing was done by project participants and a few persons within the participating institutions: LANL, CERN, UPenn (now University of Melbourne), USGS, and ODU. The original user interfaces (both publication and resource discovery) are surprisingly similar to current interfaces. Most of the changes are behind the scene or are new features. All of the new features such as server-side archivelet, export/import, persistent URLs, and validation are the result of perceived needs by the project participants. Code packages for group server (including archivelet server-side implementation) and archivelet are available in the project website shown in [17].

In the near term future we are set to field test kepler version 1.2 in four testbeds. USGS has developed a public warehouse of publications (http://infotrek.er.usgs.gov/docs/usgs_pubs/publication_warehouse_contents.html) and wants to develop a harvesting and data provider interface using the Kepler concept. The data provider end will connect to a new Journal on Natural Organic Materials that will use directly the Kepler software. The harvester interface will be a Kepler Group server that will connect to the USGS' other offices. OLAC is considering using a Kepler group server to harvest from its current collection and vice versa. Also they will experiment with the archivelets to publish field work and use the export/import feature to deliver delayed results from field studies where authors are away for prolonged periods. LANL is discussing to give researchers access to archivelets and to have "MyLibrary" [18] harvest from a kepler groupserver that aggregates the individual archivelets.

Acknowledgements: Our thanks to Herbert Van de Sompel, who suggested the idea of persistent URLs and S. Zunjarwad, who worked on the implementation of version 1.2. We wish to acknowledge our partners: Bob Wershaw and Greg Allory from the USGS, and Steven Bird from the University of Melbourne Steven Bird.

This work is supported by NSF grant 0205486.

References

[1] Carl Lagoze, Herbert Van de Sompel, Michael Nelson, and Simeon Warner, "The Open Archives Initiative Protocol for Metadata Harvesting", January, 2002. Available at http://www.openarchives.org/OAI/openarchivesprotocol.htm

[2] Kurt Maly, Mohammad Zubair, Xiaoming Liu, "Kepler - An OAI Data/Service Provider for the Individual", D-Lib Magazine 7(4), April 2001. Available at http://www.dlib.org/dlib/april01/maly/04maly.html

[3] Kurt Maly, Michael Nelson, Mohammed Zubair, Ashraf Amrou, Sathish Kothamasa, Lan Wang, and Rick Luce, "Light-Weight Communal Digital Libraries", accepted for JCDL 2004, June 2004

[4] Patrick Hochstenbach, Henry Jerez, Herbert Van de Sompel, "The OAI-PMH Static Repository and Static Repository Gateway", Proc. of the third ACM/IEEE-CS JCDL,

Houston, TX, 2003, pp. 210-217. Available at http://lib-www.lanl.gov/~herbertv/papers/jcdl2003-submitted-draft.pdf

[5] The Arc Cross Archive Searching Service, http://arc.cs.odu.edu

[6] Xiaoming Liu, Kurt Maly, Mohammad Zubair, and Michael L. Nelson, "Arc: An OAI Service Provider for Cross Archive Searching", Proceedings of the First ACM/IEEE Joint Conference on Digital Libraries, Roanoke VA, June 24-28, 2001, pp. 65-66. Available at http://www.ils.unc.edu/~mln/jcdl-arc.pdf

[7] A Digital Library that federates Physics collections with varying degrees of metadata richness, http://archon.cs.odu.edu

[8] Hesham Anan, Kurt Maly, Michael Nelson, Mohammad Zubair, Xiaoming Liu, Jinsong Gao, Jianfeng Tang and Zhao Yang, ARCHON: Building and Learning Environments Through Extended Digital Library Services, Proceedings of the 5th International Conference on New Educational Environments, Lucerne Switzerland, May 26-28, 2003.

[9] Self-Archiving and Open Archives, http://www.eprints.org/self-faq/.

[10] Anna Keller Gold, Karen S. Baker, Jean-Yves LeMeur , Kim Baldridge, "Building FLOW: federating libraries on the web", Proc. of the second ACM/IEEE-CS joint conference on Digital libraries, Portland, OR, 2002, pp. 287-288. Available at http://doi.acm.org/10.1145/544220.544286

[11] MacKenzie Smith, Mary Barton, Mick Bass, Margret Branschofsky, Greg McClellan, Dave Stuve, Robert Tansley, and Julie Harford Walker, "DSpace - an open source dynamic digital repository", D-Lib Magazine, 9(1), January 2003. Available at http://www.dlib.org/dlib/january03/smith/01smith.html.

[12] Martha L. Brogan, "A Survey of Digital Library Aggregation Services", The Digital Library Federation Council on Library and Information Resources, Washington, DC. 2003. Available at http://www.diglib.org/pubs/brogan/

[13] Raym Crow, The Case for Institutional Repositories: A SPARC Position Paper, The Scholarly Publishing & Academic Resources Coalition, Washington, DC, 2002. Available at http://www.arl.org/sparc/IR/ir.html

[14] Raym Crow, "Open Society Institute - A Guide to Institutional Repository Software", Open Society Institue, New York, NY, 2004. Available at http://www.soros.org/openaccess/software/

[15] "Jeff Young Extensible Repository Resource Locators (ERRoLs) for OAI Identifiers". Available at http://www.oclc.org/research/projects/oairesolver/default.htm

[16] Herbert Van de Sompel, Jeffrey A. Young, Thomas B. Hickey, "Using the OAI-PMH ... Differently", D-Lib Magazine, July/August, 2003. Available at http://www.dlib.org/dlib/july03/young/07young.html

[17] Kepler, http://dlib.cs.odu.edu/#kepler

[18] Di Giacomo, Mariella, Dan Mahoney, Johan Bollen, Andres Monroy-Hernandez and Cesar M. Ruiz Meraz, MyLibrary, A personalization service for digital library environments. Joint DELOS-NSF Workshop on Personalisation and Recommender Systems in Digital Libraries, Dublin, Ireland, 18-20 June, 2001. Available at http://lib-www.lanl.gov/lww/MyLibrary.pdf (http://lib-www.lanl.gov/lww/mylibweb.htm)