
Ryan Chute(1), Luydimilla Balakireva(2), Stephan Drescher(3)

Digital Library Research & Prototyping Team
Research Library
Los Alamos National Laboratory

(1) rchute@lanl.gov
(2) ludab@lanl.gov
(3) stephand@lanl.gov
Outline

Presentation breaks down into two parts:

1. aDORe djatoka Image Server
   • Introduction
   • Features & Interfaces
   • Service Framework

2. Interoperability & Integration
   • Repository: Fedora, aDORe, WWW
   • Clients: Ajax, OpenLayers, Flex, iPhone
Part 1 - aDORe djatoka Image Server
Context: The aDORe Project Design Principles

- **Scale**: Concrete need to design and implement a solution to ingest, store, access the vast and growing collection of the LANL Research Library.
- **Interoperability**: Interest in repository interoperability (OpenURL, OAI-PMH)
- **Standards**: Leverage existing standards and technologies to make development and migration more straightforward.
- **Component-based**: Use a distributed, component-based approach to meet challenges of scale.
- **Abstract Variations**: Use Digital Objects, Datastreams, and Surrogate abstractions to characterize content.
- **Uniform Access**: Facilitate a uniform manner for client applications to discover and access content objects available in a group of distributed repositories.
- **Federated Access**: Provide single repository behavior for a group of distributed repositories.
What is aDORe djatoka?

- **Open-Source**: an open-source JPEG 2000 image server and dissemination framework; provides Web Service & Java API
- **Standards**: Leverages existing standards and technologies
  - Standards: ISO JPEG 2000 / NISO OpenURL
  - APIs: ImageJ, JAI, ImageIO, OOM
- **SDK Interface**: Provides of an implementation agnostic (e.g. Kakadu, Aware, etc) framework for JPEG 2000 compression and extraction.
- **Interoperability**: Geared towards reuse through URI-addressability of all image disseminations including regions, rotations, and format transformations
- **Uniform Access**: Provides uniforms access to distributed and heterogeneous repositories using identifier resolver interface.
- **Extensible**: Provides an extensible service framework for image disseminations
Why aDORe djatoka?

- Lack of **open source** image server implementations.
- Lack of an easily **extensible** image dissemination service framework.
- Lack of **standard syntax** for the URI-addressability of image disseminations including regions, rotations, and format transformations.
- Desire to encourage the **adoption** of JPEG 2000 as a service image file format.
- Desire to develop a **community-defined** open source image dissemination server platform.
Why JPEG 2000?

• State-of-the-art compression techniques based on wavelet technology.
• **Standard**: Open Standard Specification
• **License-Free**: Implementable without payment of royalty and license fees.
• **Compression**: Mathematically Lossless, Visually Lossless & Lossy
• **Performance**: Superior compression and rotation performance
• **Multi-scale**: Multiple resolution representation
• **Random Access**: Random code-stream access and processing
• **Metadata**: Rich Metadata Support
• **Scalable**: Multiple versions can be extracted from a single compressed file.
• See project wiki for additional details.
djatoka v1.1 Key Features

Version 1.1 is now available - 2009/05/11

- **Dynamic Scaling**: Adds Scaling Support ("svc.scale")
- **JPX Layers**: Added JPX Layer Extraction support ("svc.clayer")
- **DB Resolver**: New resolver implementation (DatabaseResolver)
- **XML Box Support**: New info:lanl-repo/svc/getJP2XML Service
- **Status Check**: New info:lanl-repo/svc/ping Service
- **Access Controls** for Resolvable Referent URI (e.g., http & ftp)
- **File Support**: Wider range of possible JP2 and image file formats
- **Caching**: Improved Tile Caching Logic
- **Logging**: Improved Logging Support
- **Performance**: Improved Performance
- See [project wiki](http://wiki) for complete list.
aDORe djatoka Interfaces
Compression Features

- Application and API provide the current capabilities:
  - Default: Dynamically determines the number of resolution levels
    - # of times an image can be halved from \( \max(w,h) \) to 96 pixels or less.
  - Default values intended to provide balance of "Image Quality" and performance.
  - Highly configurable (e.g. command-line, properties file, Java Properties object)
  - Ability to define alternate JPEG 2000 compression implementation (e.g., Aware)
Extraction Features

- Application and API provide the current capabilities:
  - Resolution & Region Extraction
  - Rotation
  - Scaling
  - Support for a rich set of input/output formats (e.g. JPG, PNG, TIF, JPEG 2000)
  - Extensible interfaces to perform image transformations (e.g., watermarking)
aDORe djatoka Service Stack

Web Server
- HTTP Server

Container
- Java Application Server

Web App
- aDORe djatoka webapp

Servlet
- OpenURLServlet
  - OCLC OOM Service
  - Resolver Manager
    - IReferentResolver

Interface
- OCLC OOM Service
- IReferentResolver

Implementation
- OpenURLJP2KService
- OpenURLJP2KMetadata
- OpenURLJP2XML
- OpenURLJP2Ping
- Any Service
- SimpleListResolver
- DatabaseResolver
- AdoreRepoResolver
- Any Resolver

aDORe djatoka: An Open-Source Jpeg 2000 Image Server
Ryan Chute
Open Repositories 2009
Atlanta, GA, May 19, 2009
Why OpenURL?

• Existing solutions provide URI-addressability of specified regions, but…
  ◦ Offer limited extensibility for identifier resolution / dissemination services
  ◦ Use home grown HTTP URI syntaxes

• **Standards**: Helpful to have standardized syntax to request Regions or other services.

• **Interoperability**: Resource URIs serve the purpose of requesting services pertaining to an identified resource (the entire JPEG 2000 image), which is a very natural fit with OpenURL.

• **Extensibility**: OpenURL provides an easily extensible dissemination service framework.

• **Re-use**: Availability and familiarity with OCLC's Java OpenURL package, an open source OpenURL Service Framework.
Djatoka & OpenURL

- ContextObject contains info about a **Referent** and a **ServiceType**.
  - **Referent Identifier** (i.e. rft_id) is used to identify the resource.
  - **ServiceType** (i.e. svc_id) is used to identify the service to perform.
- Two types of Referent Identifiers are supported:
  - **Local Persistent Identifiers** (i.e. info URIs)
    - Referent Resolver implementation does the following:
      - Resolve identifier to file path or bitstream
      - Wrap resource reference in ImageRecord object.
  - **Resolvable URIs** (i.e. http, file, ftp URIs)
    - Referent Resolver implementation does the following:
      - Download remote resource & compress
      - Manage storage approach (permanently store?)
      - Wrap resource reference in ImageRecord object
- Four Service Types are currently provided.
  - Services include getRegion, getMetadata, getJP2XML, ping
aDORe djatoka Sample Service Request

Metadata Request

http://.../resolver?url_ver=Z39.88-2004
&rlt_id=info:loc-repo///00001
&svc_id=info:lani-repo/svc/getMetadata

Metadata Response

{
  "identifier": "info:loc-repo///00001",
  "width": "3103",
  "height": "4218",
  "dwtLevels": "6",
  "levels": "6",
  "compositingLayerCount": "1"
}

Region Request

http://.../resolver?url_ver=Z39.88-2004
&rlt_id=info:loc-repo///00001
&svc_id=info:lani-repo/svc/getRegion
&svc.format=image/jpeg
&svc.level=6
&svc.rotate=0
&svc.region=1024,1536,256,256

Request Response

1024

256

1536
info:lanl-repo/svc/getMetadata

- **Request** - an HTTP GET/POST request containing the following parameters:
  - **rft_id** - An unique image resource URI
  - **svc_id** - Service to be performed (i.e., info:lanl-repo/svc/getMetadata)
- **Response** - a JSON object containing the following elements:
  - **identifier** - a unique image resource URI
  - **imagefile** - local file to resource
  - **width** - image width
  - **height** - image height
  - **dwtLevels** - number of native JP2 DWT levels in image
  - **levels** - number of djatoka resolutions, using djatoka Level Logic
  - **compositingLayerCount** - number of JPX compositing layers (i.e., frames)
- **Example**

```json
{
  "identifier": "info:lanl-repo/ds/5aa182c2-c092-4596-af6e-e95d2e263de3",
  "imagefile": "/lanl/data/loc/cc5fc4f7-e50a-455f-b3ce-a6a8b54824e/WEJLNXSBCWO7LPLC7Z6DITFX7A45XR3GS.jp2",
  "width": "5120",
  "height": "3372",
  "dwtLevels": "6",
  "levels": "6",
  "compositingLayerCount": "1"
}
```
info:lanl-repo/svc/getRegion

- **Request** - an HTTP GET/POST request containing the following parameters:
  - **rft_id** - An unique image resource URI
  - **svc_id** - Service to be performed (i.e., info:lanl-repo/svc/getRegion)
    - **svc.level** - djatoka level to extract from resource
    - **svc.rotate** - Rotates image by 90/180/270 degrees clockwise. Default: 0
    - **svc.region** - Y,X,H,W. (Offsets based on max image resolution)
    - **svc.scale** - Request specific output size by scaling extracted resource.
    - **svc.format** – Mimetype to be provided as response. Default: image/jpeg
    - **svc.clayer** - Index of the JPX codestream contained in JPEG 2000 resource
  
- **Response** - an image of the mimetype specified in service request.

- **Example**

getRegion
info:lanl-repo/svc/ping

- **Request** - an HTTP GET/POST request containing the following parameters:
  - **rft_id** - An unique image resource URI
  - **svc_id** - Service to be performed (i.e., info:lanl-repo/svc/ping)
- **Response** - a JSON object containing the following elements:
  - **identifier** - a unique image resource URI
  - **status** - Indicates whether image is "OK", "PROCESSSING", or not found, as indicated by a 404 response.

- **Example**

  ```
  {
  "identifier": "info:lanl-repo/ds/5aa182c2-c092-4596-af6e-e95d2e263de3",
  "status": "OK"
  }
  ```
info:lanl-repo/svc/getJP2XML

- **Request** - an HTTP GET/POST request containing the following parameters:
  - `rft_id` - An unique image resource URI
  - `svc_id` - Service to be performed (i.e., info:lanl-repo/svc/getJP2XML)

- **Response** - a XML record with 0 or more XMLBox elements.
  - All responses are wrapped in JP2XML XMLSchema
  - Contains:
    - **JP2XML** – root element used to wrap **XMLBox** instances
    - **boxCount** – JP2XML attribute indicating number of XMLBox elements
    - **XMLBox** – unbounded element containing JP2 XML box content

- **Example**

```xml
<?xml version="1.0" encoding="UTF-8"?>
<jp2:JP2XML
  xmlns:jp2="http://library.lanl.gov/2008-11/aDORe/JP2XML/" boxCount="1">
  <jp2:XMLBox>
  </jp2:XMLBox>
</jp2:JP2XML>
```
Client Implementations

IIP Image djatoka Viewer
- Ajax-based client reference implementation
- Tile-based viewer, similar to Google Maps
- HTML / CSS / Javascript
- Asynchronous djatoka region requests
- Distributed under a GPL Free Software License

OpenLayers djatoka Viewer
- Ajax-based client reference implementation
- Tile-based viewer, similar to Google Maps
- Put an image widget on any web page
- HTML / CSS / Javascript
- Provides OpenURL Support for OpenLayers
- Asynchronous djatoka region requests
- Distributed under a BSD-style License
- Credits to Hugh Cayless (UNC Chapel Hill)
Djatoka at the Biodiversity Heritage Library

- Serving nearly 14 million pages.
- Adapted djatoka IIPImage Viewer to fit seamlessly in BHL interface
- Special Thanks to Chris Freeland, Chris Moyers, and Phil Cryer for their support and courage to be such early adopters.
- View the collection at: http://www.biodiversitylibrary.org

• Now serving all page images via djatoka (Freeland, C. & Moyers, C.)

• HOWTO: serve jpeg2000 images with a scalable infrastructure (Cryer, P.)
Technical Requirements

- Sun Java 2 Standard Edition 1.5+
- Tomcat 5+
- Ideal:
  - > 512MB RAM
  - Multiple CPUs/cores - Significant Parallel Processing Benefits
Licensing

- **Djatoka** Image Server is distributed under an **LGPL** License.
- **Kakadu** JPEG 2000 compression / extraction library
  - Free for **Non-Commercial** use
  - ~8,500 - ~35,000 USD for commercial license.
- Kakadu Binaries provided for:
  - Win32, Mac OS-X x86, Linux x86_32/64, Sparcv9
- Djatoka **IIPimage** Viewer is a modified IIPMooViewer instance distributed as Open Source under a **GPL** License.
  
  http://iipimage.sourceforge.net/

- Djatoka **OpenLayers** Viewer is a modified OpenLayers build, released under the Clear **BSD** license.
  
  http://www.github.com/hcayless/djatoka-openlayers-image-viewer
Part 2 - Interoperability & Integration
Interoperability

- Out of the box, a Resolvable URI identifier can be used to integrate/display content from nearly any repository or from the Web.
- With little effort, this interoperability approach can be integrated into your institutional repository system, such as a Fedora instance.
  - Steps necessary to integrate dja\textit{toka} with Fedora:
    - Define new Service Definition with following methods:
      - getMetadata
      - getRegion
      - getImageView
    - Define new Service Deployment for new Service Definition:
      - Map Fedora methods to dja\textit{toka} OpenURL services and service parameters.
    - Associate sDef with content model with access to datasteam.
    - Update viewer.html to use datastream URI as rft\_id.
      - var images = location.href.substring(0,location.href.indexOf("/dja\textit{toka}:jp2SDef/getImageView")) + "/source";
Djatoka OpenURL Referent Resource Resolution

Referent Resolver Process - Default

Input - rft_id
Local Persistent URI
Resolvable URI
Lookup File Path
Download
Compress
Storage
Temp Cache
Output - ImageRecord
File Path
File Path
File Path

Referent Resolver Process - Alternate

Input - rft_id
Local Persistent URI
Resolvable URI
Resolve Datastream
Download
Compress
Storage
Repository
Output - ImageRecord
Inputstream
Inputstream
File Path
Interoperability – Tighter Integration

• If you manage the resources accessed via djatoka, you’ll want to utilize a Referent Resolver instance for the following reasons:
  • By-pass need to download resource; direct access to file path.
  • No need for second cached instance managed by djatoka.
  • Rather quickly, we can integrate djatoka and aDORe with direct access to the resource.
Client Implementation Approaches

Dynamic Tiles

Window View

Full View
Current Client Implementations

- IIImage djatoka Viewer
- OpenLayers djatoka Viewer
- iPhone / iPod Touch djatoka Viewer
- Flex djatoka Viewer

aDORe djatoka: An Open-Source Jpeg 2000 Image Server
Ryan Chute
Open Repositories 2009
Atlanta, GA, May 19, 2009
Thank You

• Please feel free to contact us and thank you for your support.

• Wiki at:
  http://apps.sourceforge.net/mediawiki/djatoka/

• SourceForge effort at:
  http://sourceforge.net/projects/djatoka

• Demonstrations at:
  http://african.lanl.gov/ado-re-djatoka/