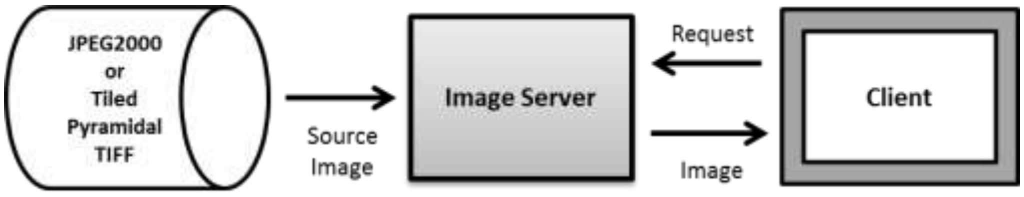
Purpose

Provide a solution for viewing high resolution still images natively in a web browser. The solution should be implemented for single image viewing, multiple image/slideshow viewing, and the page turner. This solution should be embeddable within a webpage for use in digital exhibits and other webpages.

Process of serving high-resolution images

Any high-resolution image, no matter the format, consumes megabytes of space and bandwidth to serve. In order to provide a responsive experience smaller compressed derivatives of a source image are created and sent to the client. The smaller derivatives are created based on the zoom level and location of the image being rendered. The image server, given coordinates and quality parameters, generates browser friendly PNG or JPEG from a source image file.



Release 7.7 research

For release 7.7 of RUcore some research was performed using two different image server products, IIPImage and Digilib, coupled with the OpenSeadragon image viewer. The OpenSeadragon viewer is a HTML5 image viewer that offers deep zooming capabilities. It was initially intended that the source image file format used for image viewing would be JPEG2000. Only Digilib natively supported JPEG2000, but the performance was poor. IIPImage did not natively support JPEG200 and required the Kakadu JPEG2000 library for conversion. After some discussion with the vendor the fee for using Kakadu for image viewing was not justifiable at the time.

Tiled Pyramidal TIFF

Tiled Multi-Resolution (or Tiled Pyramidal) TIFF is simply a tiled multi-page TIFF image, with each resolution stored as a separate layer within the TIFF. This is a standard TIFF extension and is supported by most image processing applications including Photoshop,GIMP, VIPS and ImageMagick. The libtiff codec library is also perfectly capable of reading and writing such images.

http://iipimage.sourceforge.net/documentation/images/

IIPImage and Digilib do natively support an alternate image format; tiled pyramidal TIFF. Tests were performed using tiled pyramidal TIFF's. Using IIPImage the performance and presentation were comparable to that of JPEG2000 samples. Digilib performance was not as good as IIPImage. The file size of the tiled pyramidal TIFF's was similar to JPEG2000 presentation test files as well.

ImageMagick can be used to generate tiled pyramidal TIFF's. The following is an example of a conversion.

```
convert source.file -define tiff:tile-geometry=256x256 -compress jpeg 'ptif:output.ptif'
```

This creates a tiled pyramidal TIFF with JPEG compression that can be used by an image server with OpenSeadragon.

During research it was found that sometimes our archival TIFF’s did not convert cleanly. To solve this problem a simple TIFF to TIFF conversion normalized the file source file that was used during the tiled pyramidal TIFF conversion.

Justification for tiled pyramidal TIFF instead of JPEG2000

In 7.7 we were only exploring JPEG2000 as a presentation format. We were not, and are not, considering replacing our archival TIFF format with JPEG2000. If we were considering using JPEG2000 as our archival and presentation format the storage cost saving would justify the expense of the Kakadu JPEG2000 library. However, since tiled pyramidal TIFF and JPEG2000 presentation files offer approximately the same storage cost and perform almost identically the added expense for the Kakadu JPEG2000 library is not justifiable.

Recommendation

It is our recommendation that we using pyramidal TIFF’s as the high resolution presentation image format with IIPImage server.

In the future if we wish to start using JPEG2000 as either the presentation format or as both presentation and archival formats the change should be straightforward. Until then, if we are required to store and create JPEG2000 for any special projects those files should only be offered as a download to the end user. Offering JPEG2000 as a file download will not incur any Kakadu JPEG2000 license costs.

Implementation – File processing

Tiled pyramidal TIFF generation should be added as part of the file processing pipeline in WMS. ImageMagick can be used to generate the tiled pyramidal TIFF presentation files using the command(s) noted above. This would serve a replacement to the JPEG presentation file generation procedure.

Implementation – Resource datastream architecture

The following table represents a resource with tiled pyramidal TIFF’s.

Datastream Identifer	**Purpose**	**Source**	**Mime-type**
ARCH1	Archival	Upload	`image/tiff`
ARCH2	Archival	Upload	`image/tiff`
…			
ARCHN	*Archival*	*Upload*	*`image/tiff`*
DARCH1	Archival	Upload	`image/tiff`
DARCH2	Archival	Upload	`image/tiff`
…			
DARCHN	*Archival*	*Upload*	*`image/tiff`*
PTIFF-1	Presentation	Generated from archival	`image/x-ptiff`
PTIFF-2	Presentation	Generated from archival	`image/x-ptiff`
…			
PTIFF-N	*Presentation*	*Generated from archival*	*`image/x-ptiff`*
PDF-1	Presentation	Generated from archival OR upload	`application/pdf`
THUMBJPEG-1	Presentation	Generated from archival	`image/jpeg`

**PTIFF datastreams are ideally generated from ARCH datastreams; however if only DARCH datastreams are provided the PTIFF datastreams should be generated from those DARCH datastreams.*

Implementation - Migration

To implement deep zooming of high resolution images we should replace the JPEG prefixed datastreams with PTIF datastreams. The file extension for the tiled pyramidal TIFF will be .ptif.

Currently, the presentation JPEG datastreams are maxed out to 1600 pixels depending on the orientation of the image. This limitation should be removed. The presentation PTIF should have the identical image resolution of the derived master archival TIFF.

All resources which have JPEG datastreams should have corresponding full resolution PTIF datastreams generated. Any new resources that are created should only have PTIF datastreams created. Presentation JPEG datastreams should no longer be an option. After conversion all presentation JPEG datastreams should be purged from the repository.

Implementation - Legacy URL Support

After purging any requests for JPEG datastreams should continue to return a JPEG as it did before, however the JPEG should be generated from the PTIF and created using the image server. This will offer legacy support for any URL's that had been bookmarked or directly referenced.

Example: https://rucore.libraries.rutgers.edu/rutgers-lib/1234/JPEG/1 should redirect to the corresponding PTIF/1 datastream and send a JPEG derivative to the client for download.

The "play" and "resize" renderers for individual JPEG datastreams should also be preserved and transmit the corresponding PTIF to the client as an inline file.

Example: https://rucore.libraries.rutgers.edu/rutgers-lib/1234/JPEG/1/play/ should redirect to the corresponding PTIF/1/play/ datastream and send a JPEG derivative to the client for inline display.

Example: https://rucore.libraries.rutgers.edu/rutgers-lib/1234/JPEG/1/resize/800x600 should redirect to the corresponding PTIF/1/play/ datastream and send a resized JPEG derivative to the client for inline display.

The "read" and "play" renderers for a resources JPEG's should be preserved and redirect the client to use the PTIF datastreams.

Example: https://rucore.libraries.rutgers.edu/rutgers-lib/1234/JPEG/play/ should redirect to PTIF/play and start a slideshow.

Example: https://rucore.libraries.rutgers.edu/rutgers-lib/44438/JPEG/read/ should redirect to PTIF/read and start the page turner.

Implementation – IIIF API Support

The International Image Interoperability Framework (IIIF) group maintains an API specification for image-based resources. The OpenSeadragon viewer can interface images for zooming and scaling using the IIIF API schema. It is our recommendation that we implement an IIIF API service point for rich image access using the OpenSeadragon viewer. More about the IIIF can be found at: http://iiif.io/

There are two major version of the IIIF specification; 1.1 and 2.0. In order to offer maximum interoperability we suggest offering support of both major versions. The following is the suggested URI structure for support.

Base - https://rucore.libraries.rutgers.edu/rutgers-lib/iiif/

Version 1.1 - https://rucore.libraries.rutgers.edu/rutgers-lib/iiif/1.1/

Version 2.0 - https://rucore.libraries.rutgers.edu/rutgers-lib/iiif/2.0/

A sample interaction with an image using the IIIF 2.0 specification would be:

https://rucore.libraries.rutgers.edu/rutgers-lib/iiif/2.0/1234:PTIF-1/full/full/0/default.jpg

This sample interaction would validate the clients access to `rutgers-lib:1234` resources `PTIF-1` datastream and return a full size JPEG file name `default.jpg`. Note a colon delimiter is used between the resource ID and the datastream ID.

The following is a more complex request for a region of an image (0,10,100,200), resized(pct:50), and rotated 90^{o}.

https://rucore.libraries.rutgers.edu/rutgers-lib/iiif/2.0/1234:PTIF-1/0,10,100,200/pct:50/90/default.jpg

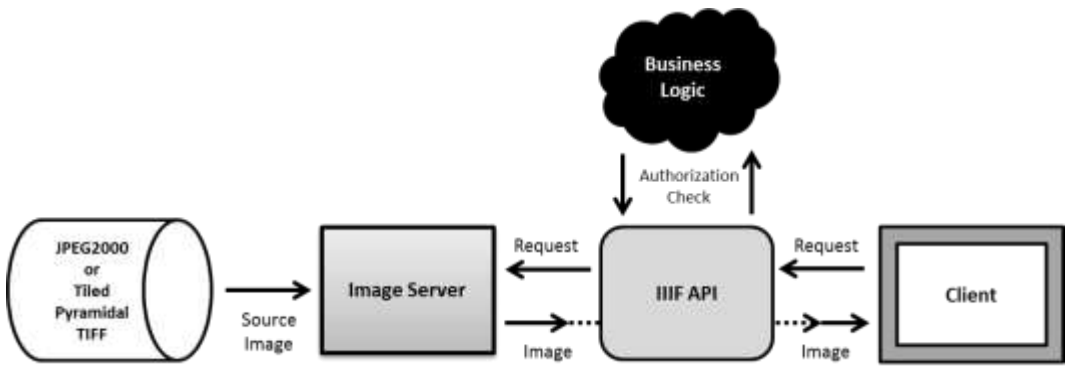
Implementation – File labeling and statistics

Currently, for JPEG datastreams if a label hasn't been defined the datastream ID is used. Once we replace the JPEG's with PTIF's the labeling may be confusing. A default PTIF datastream label should be defined that is not confusing for end users; something like "Image 1" instead of "PTIF-1."

For statistics viewing and reporting once all JPEG datastreams are replaced with PTIF datastreams the corresponding download statistics should be migrated as well. All JPEG datastream download entries should be replaced with a corresponding PTIF entry.

Implementation – Security

The IIPImage server is a Fast CGI module. When using the IIIF API support URI language described above we will be able to insert business logic for authorization checking. A client will never need to directly interact with the IIPImage server; all interactions will be managed through the IIIF API service point. The IIPImage server can then be IP restricted; localhost in our current configuration.



Technical Notes

To create tiled pyramidal TIFF’s you need to use ImageMagick 6.4.7-10 and upwards. Upgrades of ImageMagick will need to be made if approved.

During initial research a sample Bayonne map reported the following warnings:

```
convert: bayonne_map.tif: wrong data type 7 for "RichTIFFIPTC"; tag ignored. `TIFFReadDirectory'.

convert: bayonne_map.tif: unknown field with tag 282 (0x11a) encountered. `TIFFReadCustomDirectory'.

convert: bayonne_map.tif: unknown field with tag 283 (0x11b) encountered. `TIFFReadCustomDirectory'.

convert: bayonne_map.tif: unknown field with tag 296 (0x128) encountered. `TIFFReadCustomDirectory'.

convert: bayonne_map.tif: unknown field with tag 306 (0x132) encountered. `TIFFReadCustomDirectory'.

convert: bayonne_map.tif: unknown field with tag 40961 (0xa001) encountered. `TIFFReadCustomDirectory'.

convert: bayonne_map.tif: wrong data type 7 for "RichTIFFIPTC"; tag ignored. `TIFFReadDirectory'.
```

Now these are warning and should not harm conversion. They can actually be silenced using the -quiet flag during conversion; however that might silence other warning as well.