Polyglot / SoftwareServer Developer Documentation

- Installation
- User Interface
- Internal Working
  - The Java Classes
  - SS Registration
  - SS checkin
  - SS capabilities
  - Polyglot REST APIs
    - POL REST endpoints that POL handles on its own without accessing/redirecting to SSes:
    - POL REST endpoints that access or redirect to SSes:
- Implementation Details
  - Configuration files
  - Start-up / Initialization
  - kgm.utility.Utility source code
- Testing
  - Testing conversion on dap-dev
  - Bi-hourly tests.py
- Restlet Documentation
- Installation of Required Software Packages on Ubuntu Trusty (14.04)

- **Installation**
  1. On Mac, use brew to install **maven**, **imagemagick**, **rabbitmq**. Install mongo either manually or using brew.
  2. git clone the repo.
  3. In the top dir, do **"mvn compile**" to compile the code, should not take long. Then do "**mvn package -Dmaven.test.skip=true -Dmaven.javadoc.skip=true -q**".
  4. Start mongod on localhost.
  5. Start RabbitMQ on localhost.
  6. In the top dir, in one shell, run **"bin/PolyglotRestlet.sh**" to start Polyglot. In another shell tab, run **"bin/SoftwareServerRestlet.sh**" to start SoftwareServer. If the script fails, change its content from "2.1.0" to "2.2.0". Then you can use the URLs **http://localhost:8184** to check Polyglot, and **http://localhost:8182** to check SS.
- **User Interface**

A user interacts with Polyglot. Polyglot works internally with software servers. By default Polyglot runs on port 8184, Softwareserver on port 8182. A user can use "http://<polyglot_ip>:8184/" to see the available endpoints / URLs, such as "http://<polyglot_ip>:8184/servers" to see the server IP list, or "http://<polyglot_ip>:8184/form" to use a form to submit a conversion request.

- **Internal Working**
- The Java Classes
  1. Software Server:
     a. SoftwareServer.java: handles wrapper scripts;
     b. SoftwareServerRestlet.java: handles the Restlet service;
     c. SoftwareServerRESTUtilities.java: the **rabbitMQHandler()** method handles all the interaction with RabbitMQ.
  2. Polyglot:
     a. The entry point is PolyglotRestlet.java.
        `java -cp polyglot.jar:lib/* -Xmx1g edu.illinois.ncsa.isda.softwareserver.polyglot.PolyglotRestlet`
     b. Polyglot.java: abstract class;
     c. PolyglotStewardAMQ.java: handles IOGraph and interaction with RabbitMQ;
     d. PolyglotRestlet.java: handles the Restlet interface.
        process_jobs(): at the end, writes the ".url" file.
- SS Registration

A Polyglot process goes to RabbitMQ, gets the consumer IPs, connects to these IP's softwareserver at the URL "<softewareserver_ip>:8182 /applications". If the URL is accessible and contains valid content, Polyglot adds the IP to its server list.

- SS checkin

A Softwareserver connects to RabbitMQ, picks up jobs (aka msgs) in the queues, processes them, and sends the results back by accessing Polyglot's endpoint at "<polyglot_ip>:8184/checkin/<jobid>/<result_url>".

- SS capabilities

SoftwareServer uses SoftwareServer.conf + scripts/*/.aliases.txt to configure which applications it will process. For example, SS on dap-dev is configured to convert only demclip and streamclip:

SS conf and .aliases.txt content

```
SoftwareServer.conf:
#BatchScripts=scripts/bat
#ShellScripts=/home/polyglot/scripts/sh
#RScripts=scripts/R
#AHKScripts=scripts/csr-configured
#AppleScripts=scripts/applescript
#SikuliScripts=scripts/sikuli
PythonScripts=scripts/py

$ grep -v ^# */.aliases.txt
ahk/.aliases.txt:A3DReviewer
ahk/.aliases.txt:ImgMgk
ahk/.aliases.txt:IrfanView
py/.aliases.txt:demclip_convert.py
py/.aliases.txt:streamclip_convert.py
R/.aliases.txt:PEcAn
sh/.aliases.txt:avconv-audio
sh/.aliases.txt:avconv-video
```

- Polyglot REST APIs
- POL REST endpoints that POL handles on its own without accessing/redirecting to SSes:

GET:

/ Returns a list of supported endpoints.

/alive Returns "yes".

/checkin

/convert Returns all supported output formats

/convert/output_format1 returns all supported input formats that can be converted to output_format1

/convert/output_format1/file_url1 do the conversion: download file_url1

/form

/image

/inputs

/inputs/<format1>

/outputs

/requests

/servers

/software

- POL REST endpoints that access or redirect to SSes:

GET:

/file/<file1> # If file1 doesn't exist and file1.url exists.

/servers/<server1_ip>[/...] # Redirects to server1_ip:8182/software/...

/software/<sw1> # Accesses all SS:8182/software until finding one that contains sw1 and redirects to <ss_ip1>:8182/software/<sw1>.

POST:

/servers/<ip1>[/...] # Redirects to ip1:8182/software/<...>.

/software/<sw1> # Accesses all SS:8182/software until finding one that contains sw1 and redirects to <ss_ip1>:8182/software/<sw1>.

- **Implementation Details**
- Configuration files

Polyglot and SS are implemented in Java, currently using Restlet. Current configuration files are:

- Polyglot – PolyglotRestlet.conf (Polyglot port #, Softwareserver port #, etc.), PolyglotStewardAMQ.conf (RabbitMQ URI)
- Softwareserver – SoftwareServerRestlet.conf (port #, RabbitMQ URI).

Softwareserver job checkin is in SoftwareServerRESTUtilities.java, Polyglot accessing RabbitMQ part is in polyglot/PolyglotStewardAMQ.java.

- Start-up / Initialization

- When Polyglot starts, it does:
  1. read the configuration file;
  2. start the PolyglotStewardAMQ thread;
  3. start a thread to update Mongo; call PolyglotRESTUtilities.updateMongo(). By default updates every 2 sec.
  4. start the restlet service.
- When PolyglotStewardAMQ thread starts, it starts 3 threads:
  1. discoveryAMQ(), every 30 s.
  2. process_jobs(), every 3 s,
  3. heartbeat(), to remove unresponsive SSs. every Heartbeat secs, default to 10 s.
- Two queryEndpoint() methods.
  In SoftwareServerRESTUtilities.java: returns **pure text**; in PolyglotStewardAMQ.java, returns **json**.
- kgm.utility.Utility source code

  https://isda.ncsa.illinois.edu/svn/isda/trunk/kgm/Utilities/src/kgm/utility/Utility.java

- **Testing**
- Testing conversion on dap-dev
- Commands:
  - Send the conversion request:
    curl http://browndog.user:password1@dap-dev.ncsa.illinois.edu:8184/convert/jpg/http%3A%2F%2Fbrowndog.ncsa.illinois.edu%2Fexamples%2Fbrowndog.png
  - Get the converted file:
    curl -u browndog.user:password1 -O http://dap-dev.ncsa.illinois.edu:8184/file/200598567_browndog.jpg
- Used a png to test.
  http://browndog.ncsa.illinois.edu/examples/browndog.png
  The converion URL requires an escaped URL.
  To generate an escaped URL using perl:
  perl -e 'use URI::Escape; print uri_escape("http://browndog.ncsa.illinois.edu/examples/browndog.png")'

  http://www.perlhowto.com/encode_and_decode_url_strings
  man URI::Escape
- Bi-hourly tests.py
  On dap-dev in /var/www/html/dap/tests/. It used the Polyglot GET API
  using the URL to convert, got a returned URL, then downloaded that URL to
  "tmp/<count>_<file_basename>.<output_format>", then checked
  whether that file existed and was not empty.

  Python requests:
  r = requests.get(api_call, auth=(username, password),
  headers=headers, timeout=timeout)

  result = r.text
  contains only the text, not the HTML tags.
  That is, the actual content returned is:
  http://dap-dev.ncsa.illinois.edu:8184/file/200598567_browndog.jpg
  but result is only:
  http://dap-dev.ncsa.illinois.edu:8184/file/200598567_browndog.jpg

  python requests API:
  http://docs.python-requests.org/en/master/api/

- **Restlet Documentation**

  In pom.xml: org.restlet 2.3.1.

  - User guide.
    https://restlet.com/technical-resources/restlet-framework/guide/2.3

    Resource package Overview:
    https://restlet.com/technical-resources/restlet-framework/guide/2.3/core/resource/overview

    Annotation Get.
    https://restlet.com/technical-resources/restlet-framework/javadocs/2.1/jse/api/org/restlet/resource/Get.html

    Response: redirectTemporary. Used in url redirection.
    https://restlet.com/technical-resources/restlet-framework/javadocs/2.1/jse/api/org/restlet/Response.html
    https://restlet.com/technical-resources/restlet-framework/javadocs/2.1/jee/api/org/restlet/resource/ServerResource.htm

- **Installation of Required Software Packages on Ubuntu Trusty (14.04)**

Need to install the programs used in Softwareserver scripts:

convert, unoconv, daffodil, ffmpeg, 7z and 7za, libreoffice, avconv, xvfb-run, eog, flac, ps2pdf, gthumb, htmldoc, kabeja, rar (requires "multiverse"), unzip, unrar, cabextract, ncdump, pdf2djvu, prince, soundconverter, TeighaFileConverter, txt2html, unrtf, cvlc, ebook-convert

which are in the following Ubuntu packages:

imagemagick, unoconv, (daffodil/), ffmpeg, p7zip-full, libreoffice*, libav-tools, xvfb, eog, flac, gthumb, htmldoc, (kabeja?), rar, unrar, cabextract, netcdf-bin, pdf2djvu, (prince?), soundconverter, (TeighaFileConverter?), txt2html, unrtf, vlc-nox, calibre

unzip was already installed. ps2pdf is in ghostscript, already installed.

To install those that are available on Ubuntu 14.04, Daffodil and Kabeja, do:

Commands to install packages needed by Polyglot and SS

```
sudo vi /etc/apt/sources.list   # to uncomment the 4 multiverse lines.
sudo apt-get update
sudo apt-get install -y imagemagick unoconv p7zip-full libav-tools xvfb eog flac gthumb htmldoc rar unrar
cabextract netcdf-bin pdf2djvu soundconverter txt2html unrtf vlc-nox calibre

cd /home/polyglot
sudo scp -pr dap-dev.ncsa.illinois.edu:/home/polyglot/daffodil .
sudo chown -R --reference=/home/polyglot /home/polyglot/daffodil

cd /opt
sudo scp -pr dap-dev.ncsa.illinois.edu:/opt/kabeja-0.4 .
sudo chmod 755 kabeja-0.4/kabeja.sh   # Polyglot's Kabeja_convert.sh uses "./kabeja.sh", so needs to be
executable.
sudo chown -R --reference=/home/polyglot /opt/kabeja-0.4  # Kabeja_convert.sh copies the input file in it, so
needs the write permission.
```